POLITECNICO DI TORINO

Department of Electronics and Telecommunications (DET)

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

Master Degree Thesis

Video Camera Identification using PRNU pattern noise as Digital Fingerprint.

Relatori:

Candidato:

June 2016

INDEX:

Spanish Resume:

Esta tesis tratará en la continuación de otro previo que trabaja en la identificación de cámaras para investigaciones forenses, para conseguir imágenes o videos como pruebas en juicios o en dichas investigaciones. La identificación de cámaras básicamente trata en asumir un tipo de ruido como huella digital propia y única para cada cámara, así que se extrae de los videos o imágenes provistas como pruebas y se compara dicha huella con otras que están guardadas en una base de datos, donde si la coincidencia es alta se sabría a quién pertenece dicha cámara y se encontraría al culpable o implicado.

En esta tesis se considerará como huella digital en identificación de cámaras el ruido de patrón del tipo PRNU, el cual se elegirá por sus características propias, generalmente porque dicho tipo de ruido se mantiene igual en diferentes imágenes o *frames* de videos que se toman con la misma cámara, con lo cual se puede considerar como huella digital fiable. Dicho tipo de ruido proviene de imperfecciones en el proceso de manufacturación del sensor de la cámara digital, así que se ve que es único para cada cámara.

Entre otras cosas, en la tesis se proponen varios métodos de extracción de este ruido de imágenes o videos, pero el que se usará consistirá en tomar algunas imágenes como entradas, pasarlas por un filtro del cual se eliminan todos los ruidos y se obtiene las versiones idealizadas de las imágenes de entrada, y posteriormente se hace una resta entre las originales con sus correspondientes versiones ideales, con lo que se obtiene el residuo de ruido de cada imagen. Atendiendo a la característica del PRNU previamente explicada de que se mantiene prácticamente igual en diferentes imágenes de la misma cámara, se aplica un proceso de media entre todas imágenes de residuos de ruido, en el cual se obtendrá el ruido PRNU habiéndonos quitado otros tipos de ruidos indeseados como ruidos periódicos, aleatorios…

Como último paso en la identificación de cámaras quedará comparar la estimación de huella obtenida con las huellas de referencia de la base de datos, y esto se realizará mediante la operación matemática de la correlación. Dicha operación nos devolverá idealmente un 1 cuando se opera con la huella de la cámara a la que pertenece, y un 0 en el resto.

Explicando más profundamente esta introducción en la primera parte de la tesis, lo siguiente es la explicación de los diferentes códigos de Matlab realizados para llevar a cabo la mejora de los códigos realizados en el anterior trabajo de forma que detecta los fallos que tiene y propone una nueva estrategia de extracción del PRNU y la identificación de cámaras. En total hay 11 funciones y scripts de Matlab desarrollados, de los cuales cada uno es explicado tan solo lógicamente en un sub-apartado por separado. Recalco, las descripciones realizadas de los scripts y funciones desarrollados son puramente lógicas y sin atender a ninguna aplicación al tema principal de esta tesis.

El siguiente apartado se considera como el más importante, el apartado de experimentos y conclusiones. Para llevar a cabo los 5 diferentes experimentos que se van a realizar se nos proveerá con una base de datos de 13 cámaras de las cuales se nos proveerá de 13 videos de poca variación para estimar las huellas de referencia y constituir nuestra base de datos, y otros 106 videos naturales los cuales nos servirán para la identificación de cámaras.

El primer experimento consiste en una familiarización al tema, es decir, se calculan los tamaños de GOP (*Group of Pictures*, cuantos frames hay entre dos frames I consecutivos de la compresión temporal) para cada cámara mediante el uso de diferentes software como *GSpot* o *MediaInfo*. Estos datos de tamaño de GOP se guardarán para posteriores experimentos, ya que una importante premisa que se asume en la tesis anterior es que todas las cámaras usan un tamaño fijo de GOP, y veremos que esa premisa es falsa

y que será la razón más importante de los malos resultados obtenidos en este trabajo previo.

El segundo experimento consistirá en ver cuál es la secuencia real de frames que se toman para los dos primeros métodos de extracción del PRNU de los 5 que se propusieron en la tesis anterior. Estos 5 métodos son los siguientes: se toma el primer frame I para la estimación de la huella digital, se toman los 10 primeros frames I, se toman los 10 primeros frames independientemente de que tipo sean, y los primeros 10 primeros frames I y P alternativamente pero con la diferencia de que en el cuarto método se usa la medida normal y en la quinta se usa la medición ponderada (los frames I tienen el doble de peso que los P). En este segundo experimento nos centramos en los dos primeros métodos, y mediante el uso de ciertos scripts y funciones de Matlab se obtiene la secuencia real de frames que se están introduciendo para cada método en los 106 videos naturales. Se observa que se obtienen ciertas anomalías en los resultados, sobre todo en los dispositivos *Apple* que se nos han dado. En algunos videos de estos dispositivos se ve que el incremento en la secuencia de video de tamaño de GOP obtenido en el primer experimento no toma solo frames I, sino que también otros tipos de frame como el P o B. Si la premisa de la existencia de tamaño de GOP fijo fuese cierta, todos los frames de la secuencia deberían ser frames I, con lo que queda demostrado que los dispositivos *Apple* usan tamaño de GOP variable.

En el tercer experimento se hace un procedimiento análogo para el cuarto y quinto método de estimación (usan la misma secuencia de frames de entrada), para asegurarnos de si estamos cogiendo los frames adecuados en la secuencia de entrada. En resultados obtenidos se observa que las anomalías obtenidas coinciden con los videos en los que también se han obtenido anomalías en el anterior experimento, es decir, el incremento de tamaño de GOP fijo usado para la elección de frames en la secuencia de

vídeo no selecciona los frames adecuados, los que otra vez concluimos que las cámaras a las que pertenecen dichos videos usan tamaño de GOP variable, donde otra vez la mayoría de anomalías provienen de los dispositivos *Apple*.

Como hemos visto que en ciertos videos para los diferentes 5 métodos de estimación hay tamaño de GOP variable y no se seleccionan los frames deseados para la estimación, se hace uso de ciertos scripts y funciones de Matlab desarrollados donde se calcula para cada video un vector que contiene los tipos de frames en secuencia, así que sincronizando la función de estimar la huella digital con este vector hacemos que se seleccionen los frames adecuados para cada método de estimación. Haciendo esto, estimamos nuevas huellas de referencia y las naturales, esta vez habiendo usado los frames adecuados. Una vez habiendo hecho esto, mostramos para cada cámara una gráfica ROC (contiene la relación entre la tasa de detección correcta y la tasa de falsa alarma, es decir, muestra lo eficiente que es un cierto método de estimación para cierta cámara) conteniendo una curva diferente por cada uno de los diferentes métodos de estimación. Analizando los resultados se ve claramente la mejoría respecto a los resultados obtenidos en la tesis anterior, aunque en los dispositivos *Apple* resultados extraños son obtenidos, porque las curvas ROC referentes a los dos primeros métodos de estimación de PRNU, donde solo frames I son seleccionados, figuran peores detecciones que en las curvas referentes a los dos últimos métodos, donde hay tantos frames P como I, y lógicamente en los métodos donde sólo se eligen frames I debería haber mejor detección ya que estos frames no están comprimidos y transportan la información completa del ruido PRNU. Para analizar estos resultados extraños, un último experimento es propuesto para un estudio más profundo de los *Apple*.

Para el quinto y último experimento se propone la posibilidad de que el problema no venga de sólo una mala estimación de las huellas digitales naturales, sino que probablemente el problema venga de malas estimaciones de huellas de referencia correspondientes a los *Apple* que tenemos la base de datos. Así que para demostrar que el problema viene de ahí, se propone este nuevo experimento donde se tomarán de cada video de poca variación de cada cámara *Apple* dos conjuntos de frames I de donde se estimarán dos huellas de referencia para cada dispositivo de estos. Si la propuesta es falsa, es decir, si el problema no viene de malas estimaciones de las huellas de referencia, la correlación entre huellas de referencia de los dos conjuntos de frames para la misma cámara debería ser alta. Una vez habiendo computado la correlación, se obtienen valores de correlaciones muy bajos, incluso entre dos huellas de la misma cámara. Así que queda demostrado que el problema de los resultados referentes a los dispositivos *Apple* del experimento anterior viene de malas estimaciones de huellas de referencia, y como solución se propone que se tomen más de un video de poca variación para una estimación de huella de referencia más precisa.

Resumiendo, en esta tesis se han analizado las premisas, bases, programas y resultados de la tesis anterior y para ello se han analizado los orígenes de los errores que están presentes en la tesis anterior y que empeoran los resultados obtenidos para posteriormente diseñar una nueva estrategia de extracción del ruido PRNU de los videos para un óptimo algoritmo de identificación de cámaras. De esta mejora del código del algoritmo se obtienen unos resultados mejorados donde se ilustran en las curvas ROC ilustradas en la tesis, generalmente con el método de sincronización de los vectores generados con las secuencias de tipos de frames para cada video, de manera que así se eligen para cada método de estimación los frames necesarios.

1. Introduction.

As it is known, nowadays camera imaging is having a very high development in its technologies. Digital cameras have become so useful and simple that previous type of cameras, the analogical cameras, are being replaced for digital ones, especially last years for high quality digital cameras and cell-phone cameras. For this reason, image origin detection and investigation has become an issue for forensic examinations. This process is a helpful and inexpensive and mostly reliable camera identification, that could analyze an image or video as an evidence in court or in a forensic investigation to know which device has captured that video or image. But nowadays everyone has resources to manipulate and change the main characteristics of those provided images and videos so makes all this harder process, and that is the reason of trying to have a proper authentication of digital image. Only if this is available those images and videos can be reliable evidences, and this process is necessary for forensics to carry on investigations in order to use those reliable evidences in court or other justice applications.

This problem can be seen from different perspectives and approached from different ways. For example, the simplest way would be the investigation of clues in the electronical part of the evidence (the image or video). In this case those electronical files may have clues in their codification, in their headers used for transporting or compression. If those have some information of the device that has been used for recording video or taking photo, and even the camera brand of this device the investigation can be carried on. Even so, this cannot be considered as a general camera identification process, because this information is not always available. For example, if the file is re-compressed the information about the device can be removed from the header. Moreover, modifying information inside the file can be done by everyone, so this is not a reliable camera identification process.

There are some other camera identification processes, like the introduction or incrustation of invisible or even visible watermarks while the generation of images in the outputs of some devices like Epson PhotoPC 700/750Z (1.2Megapixels), Epson PhotoPC 800/800Z (2.1Megapixels), Epson PhotoPC 3000Z (3.1Megapixels) and Kodak DC290 (1). In general there are not much devices of this kind, actually there are a small number of video cameras that work in this way, but for those devices this would supply a reliable solution for camera identification problem, but as said, this cannot be applied for all type of video camera devices. Apart from this, implementing this type of technology inside video cameras increases the manufacturing economic waste. Other way to give a solution to the camera identification problem is the image noise, concretely noise that is created from imperfections on pixel values, and it is known that all imaging sensors of all video cameras have manufacturing imperfections. This thesis will be based in this way of camera identification solution. Basing in (2) it is known that those noises can be used as camera fingerprints that distinguish one camera from other, and it is explained that three different classes of noises are classified according their origin and behavior:

a) **Random or shot noise:**
   This type of noise varies image to image, it does not remain to stay the same, so it cannot be used for camera identification (2).

b) **Readout noise:**
   This noise is generated and incrusted when reading data from the imaging sensor, and even it can be seen when high brightness is in the image or at high ISO.

c) **Pattern noise:**
   The main characteristic of this type of noise is that it almost does not have any change frame to frame, so it can be considered practically as fixed along videos or images. Here can be distinguished two different types: Fixed Pattern Noise (FPN) and Photo Response Non Uniformity (PRNU).

This thesis will be focused in the third type of noise, the pattern noise. On one hand, the first classified type of noise is the FPN, and it is an additive noise that is generated by stray currents from the sensor substrate into the individual pixels known as dark currents. On the other hand there is the PRNU noise, which is generated because of the imperfections in the manufacturing of imaging sensor of a known device, which causes variation in detector size, spectral response and other imperfections as thickness on coatings. In this thesis from both types of pattern noises will be chosen the PRNU noise, because it behaves as a multiplicative factor for each pixel causing varieties in those pixel's values. This will be an advantage of PRNU against FPN, even knowing the main characteristic of both, the remaining to stay fixed along videos and images.

As said before, nowadays a big development in this ambit is happening but in camera identification its application for videos is much less investigated than its applications in images. It is known that videos can capture much more visual and dynamic information than single images, and the use of videos increased a lot mainly since cell-phones like smartphones or iPhones are available for all people. For this reason, the sharing of videos between different devices has proportionally increased too, and also the bandwidth needed for this.

Therefore, due to the advancements telecommunication networks technologies it is possible to share more and more even high quality videos. But all has its own problems, and in this case are the illegal copying and re-distribution. All those developments make this possible, for example, with small devices films can be recorded in cinemas, converted to low bit-rated and sold in black markets or distributed all over Internet. This case and other similar cases create problems to different companies and its workers. The camera identification is used by forensic detectors and investigators to detect the origin of those videos to catch who are the guilty people.

For this thesis a method introduced in a previous thesis will be examined. Although some preliminary results were obtained when this work was done, mainly centered in different effects of some different parameters, such as compression or certain frame-choosing methods, when thirteen cameras and some smooth and natural raw videos are provided for estimating PRNU noise as digital fingerprint.

From smooth videos reference fingerprints will be estimated for each camera, and those will build the database that will be used to later make cross correlation between estimated fingerprints from natural videos and the reference fingerprints of the database. This will be the most important part in camera detection. The main objective of this previous thesis was making an algorithm that detected which videos belonged to which of thirteen cameras. About one hundred natural videos were provided. In the same conditions mentioned in previous lines, this thesis will have as main aim to improve the results that are going to be analyzed in later chapters, without focusing in compression and just focusing in investigation of problems of given algorithm of mentioned thesis and try to solve them.

2. Camera Identification.

As told in the introduction, some different methods can be used for camera identification process for forensics and other different applications. The method chosen for this thesis is using the pattern noise called PRNU as camera fingerprint. Two different cases are considered for camera identification: camera identification from images and from videos.

2.1) Camera identification from images.

There are some different methods to carry on camera identification, but any of them offered a general solution for this process. So, as told in (3) and (4), there can be used image noises to consider them as camera fingerprint and there are 3 different types of noises, as explained in introduction. It is also explained that the shot noise is not applied in camera identification because it varies along frames or different images taken from the same camera. For the readout noise, which is generated by high brightness or high ISO or even hot pixels in the sensor array, but anyway it can also be considered as camera fingerprint. It is told in (5) that it can also be applied on images that have high level of JPEG compression, which means a certain loss of information for image and for that noise. But it cannot be considered as general method, because those defective pixels in the sensor array can be solved with digital image correction processes in some cameras, so this noise is not a global fingerprint for those cameras, even less for all cameras. For third type of noise, the pattern noise, is considered the best option for global camera identification. As explained in introduction there are two pattern noises, one is the Fixed Pattern Noise (FPN) and the other is the Photo Response Non-Uniformity (PRNU). The first type is considered as dark current noise, which only can be extracted from dark images and its main problem is that it is compensated with sensor calibration.

So for this reasons taking the FPN as fingerprint is cannot be considered as global camera identification process, because is just limited to dark images and it is additive noise. In the next subchapter PRNU pattern noise will be explained and its advantages against FPN pattern noise.

2.1.1) PRNU pattern noise.

PRNU pattern noise come from imperfections and defects in the manufacturing process of the image sensor of the camera. So let's explain this process step by step to see the origin of this type of noise.

This mentioned imaging sensor is basically composed by lots and lots of little elements called pixels, whose main task is the conversion of the receiving light from the scene (image or frame captured with the camera) into electrical voltages, always depending on the quantity of light they receive. But first of all antialiasing filter is applied on the incident light and some frequencies from this incident light's spectrum are chosen by using the CFA (color filter array) to only take basic colors to reach the pixels. So for this step an analogical signal is obtained. This process is explained in (6) and (7), there also can be said as extra information related to this that in the early past analog sensor for the visible light spectrum part were video camera tubes, but nowadays there are used CCD (semiconductor charge-coupled devices) or CMOS (active pixel sensors in complementary metal-oxide semiconductors) technologies. The next step is to convert this analogical signal into a digital signal, which this task belongs to the A/D converter (Analogic → Digital). There also is needed to say that for digital cameras there are other tools and processes to manipulate or correct the image, such as Gamma correction, white balance or color adjustment and correction. In the image 2.1.1.1 there is a visual summary of this scheme.

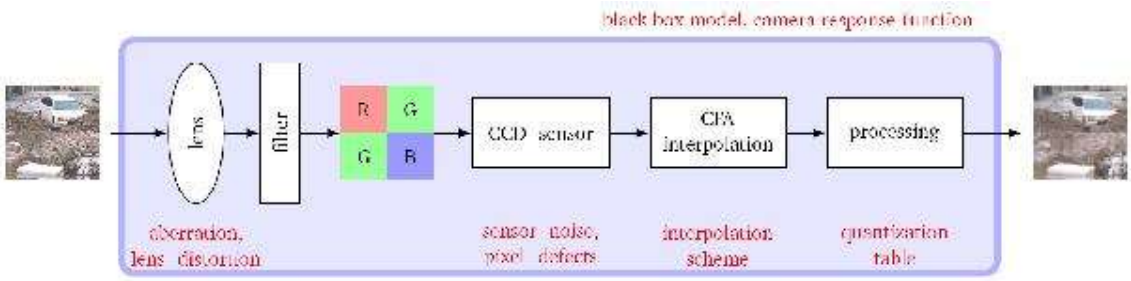

Image 2.1.1.1. Digital image acquisition process.

Once explained this, there can be better understand the image acquisition process where some different noises are introduced in the output image during this process, although the pattern noise is the only noise that can be considered as reliable fingerprint for camera identification. Remembering the fact that pattern noise is reminding almost the same along videos or between different images taken with the same the camera, we can obtain this fingerprint applying average procedures, because the part of pixel values that belong to images will be compensated and almost turned to null after averaging, and the part of pixel values that belong to PRNU pattern noise will only stay after average, because it remains to stay almost the same in different frames. With this averaging we also eliminate some periodic and random noises, so this average process is a good process to just only extract the PRNU fingerprint.

Even in smooth and flat videos there will be PRNU pattern noise, because different pixels have many sensitivity levels to light due to imperfections in the imaging sensor manufacturing process (actually for that reason will be more reliable obtaining it from those videos). This is called Pixel Non-Uniformity (PNU). For this the PRNU pattern noise is unique for each camera, even if they are manufactured in the same way and they are the same model of camera. PRNU pattern noise is a strong and robust noise that can be considered as a reliable fingerprint, and it is robust against natural factors such as temperature.

But not only are imperfections in the manufacturing of the imaging sensor. The imperfections in the lens (both in its optical surface and zoom settings) and even powder particles on it are factors that develop the PRNU generation in the output image, apart from the own PNU of the camera, generating mainly known as “doughnut” patterns or low frequency components. Those patterns do not belong to PRNU pattern noise, and they are easily eliminated by applying the average method previously mentioned.

So the main reasons of taking PRNU as fingerprint for camera identification are:

1) Unlike the fixed-pattern-noise, it can be considered as a pixel-per-pixel multiplicative factor.

2) It is robust against natural factor that can harm the output image.

3) It can be estimated applying averaging processes, taking out other noises like random noises, periodic noises or “doughnut” patterns.

4) It is unique for each camera because it comes from imperfections in the manufacturing processes of the image sensing array of the camera, so the PRNU introduced in one frame is the same in other frames taken with the same camera.

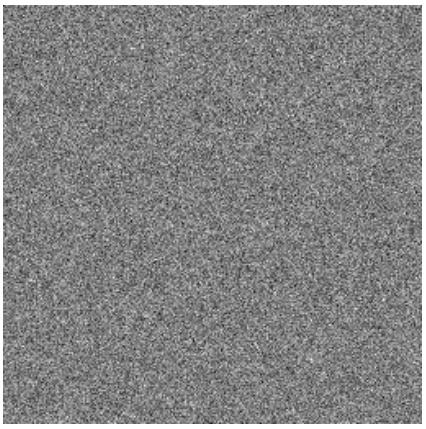

Image 2.1.1.2. PRNU pattern noise example.

2.1.2) PRNU Fingerprint Estimation process from Images.

Once we have established why it is better considering PRNU noise as a camera fingerprint, the next step will be to extract it from the acquired images (8).

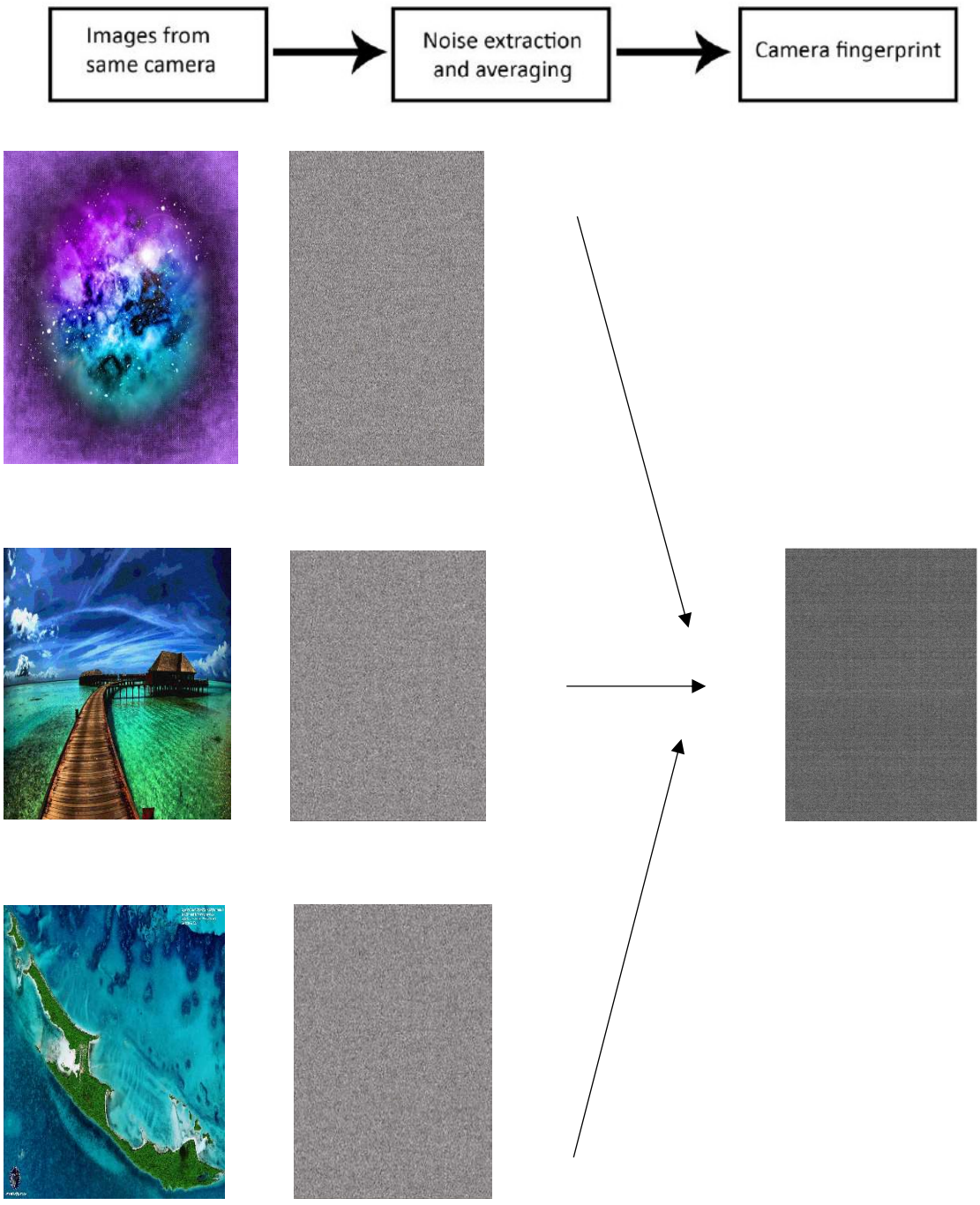

Image 2.1.2.1. Resume of PRNU fingerprint estimation.

The full process of estimating PRNU fingerprint is the next. First, all images that are taken with a camera to estimate its fingerprint need to be stripped out of noises, whose task is of the denoising filter. PRNU noise is retained by averaging those extracted noises where other types of noises like periodical noise are aliminated. So the denoised versions of the images that are provided are got or in other words, the "ideal" version of themselves. In the next equation all this expressions take algebraic meaning (9):

$$O = g^{\gamma} * [(1 + K) * I + Z]^{\gamma} + Q \qquad \text{(Eq. 2.1.2.1)}$$

Parameter meanings:

- γ: Gamma correction factor, typically with value of $\gamma \approx 1/2{,}2$.
- g: Color channel gain (different for each channel).
- I: Intensity of light.
- Z, Q: Noise sources: dark current (FPN), shot noise, readout noise (all those noises englobed in Z), and quantization noise (origin in lossy compression, Q).
- O: Output image with all its components.
- K: PRNU multiplicative and zero-mean factor.

Considering that in the expression $[(1 + K) * I + Z]^{\gamma}$ the most relevant factor is the intensity of light versus the expression in brackets, there is the possibility to factor the light intensity out and develop the Taylor's series or expansion, and just taking only the first order the equation 2.1.2.1 is going to be changed to the next expression:

Taylor's expansion for first order: $(1 + \mathrm{x})^{\gamma} \cong 1 + \gamma \mathrm{x}$ (Eq. 2.1.2.2.)

$$O = (\mathrm{g} * \mathrm{Y})^{\gamma} + K * (\mathrm{g} * \mathrm{Y})^{\gamma} + W \qquad \text{(Eq.2.1.2.3.)}$$

Where:

- $1 * (g * Y)^{\gamma}$: The noise-free version of the output (passed from denoising filter), let's call it as $^{(0)}$.

- K: Is the same multiplicative factor of the PRNU pattern noise as in the equation 2.1.2.1, with zero-mean.

- W: is a complex parameter that is totally related to the noises and parameters that are completely independent from denoised image and the PRNU noise. This englobes all noise sources previously mentioned (readout noise, dark current …).

Once we have this, the next step will be to denoise the images and make the subtraction between the image and its denoised version to only have the PRNU pattern noise plus other random noises and periodic noises. The next step will be to apply on those the averaging process previously mentioned and described with all of those subtracted images. Once doing this, the other noises will be eliminated and only will stay the PRNU pattern noise considered as the fingerprint for this camera to camera identification, which this process will be explained in the next chapter.

So let's mathematize this last process of denoising and averaging. Assuming M rows and N columns (M*N pixels), for the k-th input image the noise residual that is going to be averaged the correct expression will be:

$$W^{(k)} = O^{(k)} - O_{denoised}^{(k)} \qquad \text{(Eq. 2.1.2.4)}$$

Where:

- O $^{(k)}$: k-th image.
- O $_{\text{denoised}}$ $^{(k)}$: O $^{(k)}$ passed from denoising filter, ideal version of it.
- W $^{(k)}$: Noise residual for the k-th image, yet with random and periodical noises.

For averaging, the following equation explains the procedure:

$$\widetilde{W}(i,j) = \frac{1}{L} * \sum_{k=1}^{L} W^{(k)}(i,j); \quad 1 \leq i \leq M, \;\; 1 \leq j \leq N \quad \text{(Eq. 2.1.2.5)}$$

Where:

- $\widetilde{W}(i,j)$: The average of noise residuals calculated pixel per pixel (remembering that this noise residual is the subtraction between original image and its denoised version) for each input image provided.
- L: Total number of input videos provided for fingerprint estimation.
- $W^{(k)}(i,j)$: Noise residual pixel per pixel for one single camera (subtraction).
- M, N: Number of rows and columns of images, respectively (total number of pixels are $M * N$).

To end with this chapter there we have to say that those approaches that are previously commented by equations are supposed for grayscale images. If there are colored images the estimation has to be done separately for each channel proportionally with the next relation:

$$Y = 0.3R + 0.6G + 0.1B \qquad \text{(Eq. 2.1.2.6)}$$

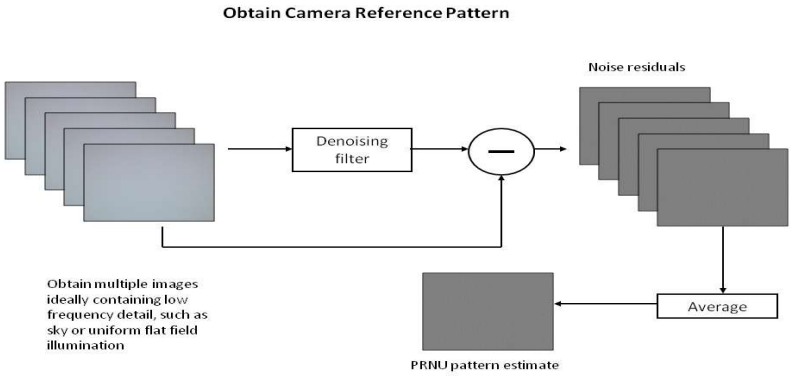

Image 2.1.2.2. Summary of the fingerprint estimation process.

2.1.3) Camera Identification for Images using PRNU.

The basic idea of camera identification is to detect de source camera of a given image or video. If consider the method of taking PRNU as fingerprint, this idea becomes to estimate fingerprint of those mentioned inputs and compare them with some fingerprints that are available in a given database. Those fingerprints that are called Reference Fingerprints, and they usually are estimated from flat and smooth frames (low variance) so the estimation of this reference is more accurate.

So, the process of comparison is carried out by using the correlation mathematic operation, which analyzes how similar is one input signal to another different signal (2). This will always return a value between 0 and 1, those values included. Mentioned correlation's application in camera identification using PRNU works as the higher value is returning the process, the more similar are both input fingerprints. So for camera identification it will ideally suppose that the correlations of a fingerprint with all of the reference fingerprints of database will return in all 0 except in one, in which will return the value 1 and in which it means that the input image or video belongs to that camera of the reference fingerprint of database.

The mathematic definition of correlation between two vector subspaces X, Y ∈ R^{M*N} is the next (9) ($\bar{X}$ and $\bar{Y}$ are the means of both subspaces):

$$Corr(X,Y) = \frac{(X-\bar{X}) \odot (Y-\bar{Y})}{\|X-\bar{X}\| * \|Y-\bar{Y}\|} \quad \text{(Eq. 2.1.3.1.)}$$

Where:

- $\odot, \|Z\|$: Is the dot product operator, that makes the multiplication point-per-point of elements of two vectors:

$$X \odot Y = \sum_{i=1}^{M} \sum_{j=1}^{N} X[i,j] * Y[i,j] \quad \text{(Eq. 2.1.3.2.)}$$

$$\|Z\| \ (norm\ of\ Z) = \sqrt{Z \odot \bar{Z}} \quad \text{(Eq. 2.1.3.3.)}$$

2.2) Camera Identification from videos.

As mentioned in the introduction camera identification in videos is much less studied than in images. The main difference between videos and images is the compression they have on themselves. Images do not need much compression because they do not have too much size, so little compression is applied on them. In the case of videos much more compression is applied to reduce their size to better and easier share videos in networks. Some of those compression standards are MPEG-1, MPEG-2, MPEG-4 (different protocols such as AVC…) and H.264, where last two mentioned are now getting stronger because previous standards are getting old and obsolete versus newer versions like those last two. But this compression carries with itself a big problem: the more compression is applied, the more information loss is and the less accurate will be the fingerprint estimation. Some different phases of compression, such as quantization directly affect in fingerprint estimation in video, because the bigger is the quantization step the bigger is the error that this introduces, so this has much more loss of information (and in this information that is lost maybe is information about PRNU fingerprint).

So in this chapter we will first explain how video compression works, then we will show how this affects PRNU fingerprint estimation in videos.

2.2.1) Video Compression theoretical basis.

Main objective of video compression is to reduce the memory space needed for a video to easily sharing, as said in the previous chapter, and many other applications, by exploiting different type of redundancies. So, the first type of redundancies for video compression are the temporal redundancies. Dividing each frame in macroblocks of same size, a reference image is taken and analyzed the chronologically next frame. Consists in analyzing how the part of image that contains each macroblock varies from the reference image to the analyzed one. As an input there is provided a raw video, and taking one (or two) frame as a reference frame and taking other frame chronologically later there is made a subtraction between both frames, and a difference frame obtained, where this is going to be sent to the network. Notice that if the analyzed frame varies slightly from the reference one, a lot of information will be taken out. As an extra information, from the reference frame to the analyzed one are used some processes like “Block matching” technique or any other to calculate motion vectors, to in the reception reconstruct the original image doing the motion compensation. For this, so, are some different frames classified along video:

- Intra or I-frames: images without any compression, all pixels of all macroblocks are encoded, and there is only one for each GOP (Group of Pictures) and also it is used as reference frame for next types of frames.

- Predicted or P-frames: images with temporal compression applied as explained before, but only just taking the previous I or P-frame as reference frame. So, less information than I-frames have those frames.

- Bidirectional or B-frames: images with temporal compression applied, but having two references: the chronologically previous I or P-frame, and the other the chronologically next I or P-frame. This type of frame is the one which has most compression.

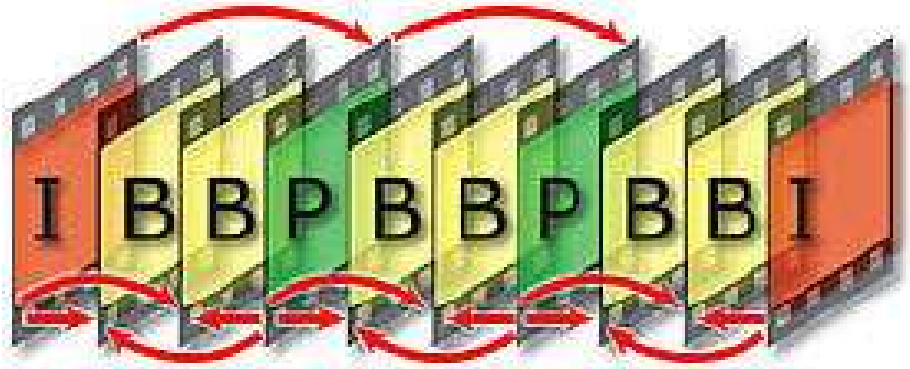

Image 2.2.1.1. Different types of frames along a video with their references illustrated with red arrows.

The next type of mentioned redundancies are the spatial redundancies. This refers to redundancies in frames that have similar intensity values in adjacent pixels in different zones of the frame, for example, in an image with a face and a nature background in the zone of the frame that belongs to the face the pixels have very similar values along the skin parts of the face, or at least very similar values, so the spatial redundancy exploiting compression consists in encoding those pixels with less bits in order to have less information in the video. This spatial redundancies exploiting compression is applied in frames where all pixels encoded, without any other compression, so here can be said that this compression is only applied in intra or I-frames, frames with all pixels of all macroblcks encoded as explained before.

The next type of redundancies are the perceptual redundancies. Cameras capture a lot of information that human eye is not able to perceive, especially very high frequencies (psychovisual redundancy). All this information that is not needed is captured by the camera, so with this type of redundancy exploiting compression let to suppress al this information without harming the global quality of the images.

The last type of redundancies would be the entropic or statistic, based in entropic codification. This consist in giving shorter codes to those codes which have more probability to appear in the same image, in that way there is not any waste of memory.

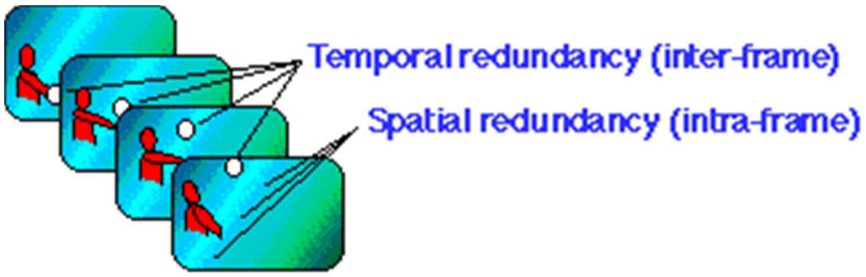

Image 2.2.1.2. Differences between temporal and spatial redundancies graphically.

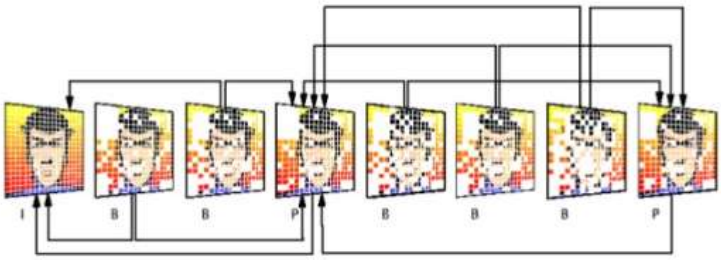

Image 2.2.1.3. Progress of information along temporal redundancy exploiting compression.

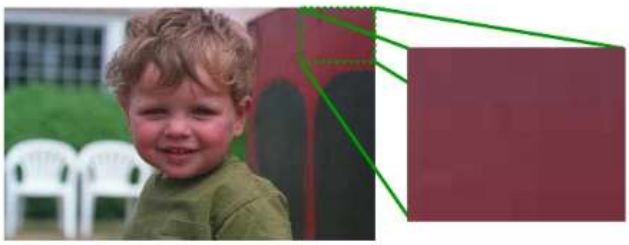

Image 2.2.1.4. Spatial redundancy exploiting compression possible application.

2.2.2) PRNU Estimation from videos.

Camera fingerprint estimation from videos is similar to the process of the estimation of it from images previously explained, because a video is a continuous succession of images or, in this case, frames. However there are some differences. Apart from the difference described in the introduction of this chapter, the compression way, there is another factor that influences differently in fingerprint estimation in videos comparing to images and it is totally related to compression. It is that the images have temporal redundancy exploiting compression on themselves, and as it is well illustrated in the image 2.2.1.3, in a same GOP there is a loss of information which is the main objective of compression. But the problem is that in this process the loss of information that carries on takes out a big part of the useful information about fingerprint. So this has a main conclusion: for a proper fingerprint estimation for camera identification from videos is better to use intra or I-frames, which do not have any compression on themselves, than using other type of frames like P-frames, and even more than using B-frames, which last one frames are more comprised than other because of the fact that they have two references while P-frames have just one.

As it is explained in the introduction of the thesis, the use of cell-phones like smartphones or IPhones for video sharing is getting more and more increased, and to make this possible networks have to be as wide as possible and videos have to be as "light" as possible (talking about memory needed) to make this easier. So videos recorded with cell phones need to have high ratio of compression, which makes P-frames and B-frames even less reliable for fingerprint estimation in those cameras of cell-phones. Also there are some parameters of the compression that get even lower the reliability of those frames, for example as it is previously said, the quantization. The bigger is the quantization step, the bigger is the error that it carries above and the bigger is the loss of information, so in those frames that are compressed this loss higher and their reliability goes lower.

So for mathematizing this estimation process there is convenient to simplify this to some approaches where we consider a video as a continuous succession of images and the first and simplest uses the next equation that is similar to the same process of fingerprint estimation for images, where all frames of the video are taken:

$$K = \frac{\sum_{i=1}^{N} W^{(i)} * O^{(i)}}{\sum_{i=1}^{N} (O^{(i)})^2} \qquad \text{(Eq. 2.2.2.1.)}$$

Where:

- K: Factor related straightly to PRNU fingerprint of camera.
- $W^{(i)}$: Noise residual of this frame (subtraction between frame and its denoised version).
- $O^{(i)}$: Corresponding frame from video.
- N: Total number of frames along the video.

As second approach there would be the one that only takes first I-frame for each video (normally the first frame in all video), because there is previously concluded that those frames are those that carry most information about PRNU fingerprint and it is interesting only analyzing those frames for estimation of fingerprint. The main problem is that if many videos of the same camera are not provided there will be only a little quantity of I-frames for estimation so this will be low accurate. Otherwise if more videos are provided for estimation the total number of I-frames would be increased, and more accurate would be (the main reason is that I-frames of different videos taken with the same camera have low correlation between them, so the different information provided for fingerprint estimation is more varied in order to have a more accurate result). This is summarized in the next equation:

$$K = \frac{\sum_{i=p(i)}^{N} W^{(p(i))} * O^{(p(i))}}{\sum_{i=p(i)}^{N} (O^{(p(i))})^2} \qquad \text{(Eq. 2.2.2.2.)}$$

Where:

- $K, W^{(i)}, O^{(i)}, N$: Same parameters of the equation 2.2.2.1, PRNU factor, noise residual for correspondent frame, the correspondent frame and the total number of frames in video, respectively.

- $p(i)$: Vector which contains the index numbers of I-frames in the current analyzing video, just to make sure that correct frames are going to be taken for estimation.

As a final example of different approaches for fingerprint estimation from video, there would be the approach of taking only I-frames for estimation (is different from previous one because digital cameras do not have to have as first frame one I-frame, it is just a common occurrence). As general rule, I-frames are more reliable than compressed frames like P-frames and B-frames, but the reliability of I-frames varies, mostly depending of type of camera. Mainly mobile phone cameras use so low bit rate when encoding which implies that there are as much compressed frames used as possible and the total number of I-frames is very low, and as explained before, the less I-frames are provided the less accurate will be the estimated fingerprint. Apart from that, normally the videos are shared in networks are short so the quantity of I-frames probably will be even lower if both problems happen in the same video. In those cases there is the need to give a solution. One of those solution is to take advantage of the P-frames that are in the compressed making a “weighted” measurement. This means that for I-frames there is given more weight than P-frames because they carry more information, but P-frames also carry some usable information of PRNU fingerprint so this would be a mixed method taking as much I-frames as possible and also some P-frames to have more information for estimating an accurate fingerprint or at least a proper approach of it.

3. Description of used MATLAB code.

Several MATLAB codes are used to carry on this process of investigation of problems in previous thesis algorithm. In this chapter only the used code will be explained and after that we will see its applications in the experiments. Exactly 11 different executable MATLAB scripts and 3 main functions, and each one will be explained to make understand what they exactly do.

First of all two different groups are distinguished in those MATLAB codes: old version codes provided from previous thesis and generated in this thesis to improve this older versions. All of them will be explained in the next two subchapters.

3.1) Old codes given for this thesis for first experiments.

This group is made by two executable MATLAB scripts and one main function which is going to be used in those executable codes. There are more pieces of code given of previous thesis but they are going to be modified and explained in the next subchapter. Those all codes are executed and tried in the beginning to get familiar with this task and to see where they can be improved to get better results of camera identification, and we will see in chapter 4 that this is going to happen.

3.1.1) GetFingerprint_video_arrayEdited.m

This function is explained in this group of code although it is used in the improved versions of the obsolete codes because it is not changed, it is used as it is provided at the beginning. Its main function is to estimate one fingerprint from all videos that belong to a directory that is putted as input parameter.

It makes for each video all the process described previously (makes the subtraction of all frames with their denoised versions and after that the average), and it returns a matrix of values of 3 dimensions (taking in account the color channels for videos with color). Apart from the directory of the videos are wanted to estimate their Fingerprints there are other inputs, like a vector containing the number of frames are wanted to take from video for estimation, the GOP size of the video, and a boolean input which controls the "weighted" measurement (0 if there is wanted to be a no-weighted measurement and 1 for the opposite).

So for Natural Fingerprints should be created a different folder for each video, because this function takes all videos from a directory and brings back one output Fingerprint, and if we want to estimate one single Fingerprint per video using this code there each video should be in different folder. Once done that, the way to make different type of measurements is varying the input vector that contains which frames are used for estimating the fingerprint (for example if the Fingerprint is estimated from the first frame the input parameter should have value [1], and if the fingerprint is estimated from the first two frames it should have value [1 2]).

3.1.2) VIDEO_FingerprintEstimation.m

In this executable script 5 different ways are proposed to estimate the Fingerprint with 5 different combinations of frames. In this script the previous function will be used for estimation using the directory configuration explained of having one video per different folder, and the estimation will vary changing the input frame index containing vector. Once the previous function is being applied for any of the 5 different ways of estimation on a loop of 106 repetitions, while an output Fingerprint is brought back from the main previous function it will be passed from an RGB to greyscale converter and this Fingerprint will be saved in the current directory, for each repetition of the loop.

This process will automatically generate 106 different variables for each method is wanted, 5 in this case. For each method the input frame index containing vector will have to be changed manually. Also there is needed to explain that there is one vector of size 106 that contain the GOP sizes of each video and one of its values is applied correspondently to the video is analyzing for each time of the loop.

3.1.3) VIDEO_RefFingerprintEstimation.m

In this executable script the procedure is very similar, but in this case there is no loop. All input parameters for the function are manually introduced for each camera of 13 are provided. The output variable will be saved in the current directory and its name will be manually changed. The rest of the process is similar to the previous script.

3.2) Codes are used for experiments of this thesis.

This group of codes is formed by two new designed functions and 9 other MATLAB executable scripts with each one does different tasks. All of them are going to be explained with details of how some of them are programmed, and how others are improved from their older versions.

3.2.1) Get_Frame_Info.m

This function is used to take as input the path or directory of a certain single video, and one number that is embedded in the title of the output information text file created by using a certain tool in order to distinguish it from other text files with similar titles. This function takes the input video, and uses the mentioned tool called *ffprobe* of the software *ffmpeg*, which takes an input video and brings back an information text file with some information for each frame of the input video, including which type of frame.

To execute this tool without a graphical interface there is needed to execute the next command line:

```
ffprobe -show_frames -select_streams v:0 video_dir > output_file_dir.txt
```

This function is very useful to later check the full structure of frames along the input video, because in some videos there is not fixed GOP as it is going to see in the next experiments of the chapter 4, even some software tell that the GOP size of the video is of a certain value.

3.2.2) Get_Frame_Array.m

In this similar function we have as input one input parameter the path where a certain information text file generated with the previous function is located, and as other input parameter a certain number to distinguish the output variable from others can be generated, normally this number being in line with the number of the input information text file.

In this function all text file is read and saved in a string variable, and it is divided in some string pieces splitting where the sentence 'pict_type' is wrote because it is unique for each frame. Once done that, in each piece of string are found the sentences '=I', '=P' or '=B', and depending which one is found a certain value is saved in the output cell-array. This is applied to all the string array, and as a result a vector type cell-array is generated containing the type of frame that contains the video, in chronological order.

This generated output will be very useful to check which frames are taken for any estimation way, and as improvement to make sure that correct frames are taken for estimation.

3.2.3) VIDEO_GetFrameInfo_File_and_Array.m

This executable MATLAB script contains both functions previously described in a loop of 106 repetitions (one for each natural video), where in each repetition is generated using the first function a frame-information text file for the correspondent video, and taking this file as input for the second function there will be generated a cell-array variable which contain each type of frame for the input video. Finally this generated cell-array will be saved in the current directory and to distinguish each one from others it the input number for each function is used.

In general, once all script is executed 106 information text files and correspondently 106 frame array variables will be generated and those variables are going to suppose the key for the improvement of older codes programmed in previous thesis works, because they will be the references for the check of correct frames are taken or not.

3.2.4) VIDEO_CheckFrames_FirstMethod.m, …_SecondMethod.m and …_Fourth_FifthMethod.m

Those 3 executable MATLAB pieces of code are used to check if correct frames are taken for the different ways of estimation are going to be proposed in next experiments and are already proposed in previous works, concretely there are 5 methods but one of them do not have any sense checking the correct frames. So for this reason only four of them will be checked, and in the experimental chapter it will be explained.

The 3 of them work very similarly. First of all one text file is generated of a certain name (each name correspondent to the Fingerprint estimation way is being analyzed) and after that all calculated variables using the previous script are loaded in another loop of 106 times, one for each video.

So while the loop is working, in a single time of that loop from the just loaded frame array some frame indexes are taken, just the same frames that are taken for Fingerprint estimation using the correspondent method in the old codes. For example, if in those previous thesis codes is taken the first frame of the video, the first frame of the loaded variable is going to be taken. So when this or those frames are taken they are going to be written in the output text file in order to see which one is exactly the taken frame sentence. They automatically generate those text files, and according to the theoretically frame sentence should be taken for estimation, in this text anomalies are easily detected. Those text files generated by mentioned scripts are very useful tools to verify some wrong ideas are assumed in previous thesis and also to make sure in which cameras a correct frame sentence is taken and in which of them do not.

3.2.5) VIDEO_GetFingerprint_CorrectFrames.m

This process is very similar to the older versions on this code is based, actually it uses the same functions that those older scripts do, but there is a key difference between them: here in each time of the loop of 106 repetitions its correspondent frame array variable is loaded containing all frames it has generated with the *VIDEO_GetFrameInfo_File_and_Array.m* MATLAB script already described.

Once this variable is loaded, for each method of estimation is chosen the last step would be choosing the frames are wanted from the loaded cell-array. For example, if only I-frames are wanted is enough to apply the *find* instruction to find all I-frames are in that cell-array and it brings back an array with the indexes of those wanted frames in the video. After that from this index vector can be chosen all frames are necessary for the chosen Fingerprint estimation method and this vector is putted as input in the *GetFingerprint_vide_arrayEdited.m* function, where only the wanted frames will be taken for the estimation.

The rest of the process is the same as explained in the 3.1.2 subchapter. So in this way we make sure that correct frames are taken without basing in any supposition or premise. This is the biggest improvement from older versions to those new versions.

3.2.6) VIDEO_GetRefFingerprint_CorrectFrames.m

This script has, as the previous code, straight relation with its correspondent older version mainly because it is based in that older MATLAB code described in subchapter 3.1.3 and it is developed in this new version applying the same change applied in the previous program.

First of all the first change is applied is just one that makes this Reference Fingerprint estimation much more comfortable process, and it is that it generates all those estimations automatically without having to change the input video directories manually and also the output variable names. But the main change is that here also correct frames are chosen, previously generating the video information text files and the frame arrays using the function previously described which does those tasks.

For estimating those Reference Fingerprints the strategy that is proposed is taking the first 20 I-frames, so same process is applied of previous code: correspondent frame array variable is loaded, I-frames are found using *find* instruction which brings back a vector containing their indexes in the video, and only the first 20 are saved in the vector which is going to be the Fingerprint estimator function input. Here we see that in general has the same function of the code described in 3.1.3, but with applying those changes that are described in those lines and make the a code better and more comfortable Reference Fingerprint estimator.

3.2.7) VIDEO_CorrUncompressed.m

Although this script is almost the same as the one is provided from older thesis works, it is explained in this chapter because it has slight changes that make this script little more different from the mentioned old one. Here are two different ways to make the correlation between previously estimated both Natural and Reference Fingerprints, because to make the correlation processes with Fingerprints estimated by some methods of the mentioned 5 there is no unique way to do those correlations for all estimation methods so there is needed to design some alternative ways to do this process for that methods. This will be better explained in the experimental part that comes next to this chapters where those 5 different methods will be explained.

So this script loads for each Reference Fingerprint all Natural Fingerprints estimated and makes the correlation using the specific function that does it. Also there is a function that crops both Fingerprints taken for one of the correlation operations in case of one of them have different size, because to compute this operation both Fingerprints must have same number of points or, in this case, pixels.

After all this the one two-dimensional array is obtained, which includes all values of all correlation operation made. This array type is known as Confusion Matrix and it shows, as said, all results obtained for each estimation method and it is very useful to check results easily.

3.2.8) VIDEO_CorrelatedArrayTheresholding.m

As it is known, the standard deviation of a correlation process has directly relation with the number of pixels of a certain Fingerprint are being correlated, and it has the next mathematical notation where N is supposed as the total number of pixels that are being taken for correlation from a Fingerprint:

$$stddev = {}^{1}/_{\sqrt{N}} \qquad \text{(Eq. 3.2.5.1.)}$$

The main application of this standard deviation in camera identification is that most typical threshold taken to consider higher correlation values than this threshold as well detected and lower values as fingerprints of videos that do not belong to a certain camera, is three times this standard deviation that it does not have to remind to stay the same for all correlation processes. Actually, it depends in the number of pixels are being taken for correlation, as previously seen. So in this script calculates the same confusion matrix that obtains the previous MATAB code, but in this case this described threshold based in the standard deviation is applied, being this calculated for each correlation process. The output will be a binary confusion matrix where the “1” values mean that in those cases correlation values obtained are higher than the estimated threshold in that case, and the “0” values mean just the opposite case.

This script gives us a purely illustrative output, just to better see in which cameras are still being anomalies, because in previous newer versions of Fingerprint estimation programs we are sure about the correct frames are taken so the method check script will not help checking this. The anomalies detected in this binary confusion matrix can be further studied once they are detected to investigate their origin.

3.2.9) VIDEO_PLOTS_Correct.m

This script also is based in other provided at the beginning of the project but as the last explained script, this also has slight changes in its design. The main function of this code is to generate a type of graphic that illustrates how efficient is an estimation method in camera detection by showing the relation between Detection Rate and False Alarm Rate, called ROC Curve. This type of graphic is going to be explained in the nest chapter where also is going to better see the application of this statistical graphic in camera identification.

So to generate those graphics several mathematic operations are done (they also are in the older versions of the code) and up to this point there is no change from previous versions, except the new and correct correlation values obtained from correct estimations of both Natural and Reference Fingerprints. From here there are two similar ways of finishing the code: one shows for some cameras all 5 method ROC Curves in the same plot, and in for other cameras only 3 ROC Curves because in that cameras 2 methods are not being implemented (the reason will be explained in the experimental chapter).

This script will automatically generate one plot for each camera containing 3 or 5 ROC Curves, depending to which camera they belong. Once this is done, there can be deduced that those graphics are the best way to show how good the created camera identification algorithm is.

4. Experiments and Results.

After the description of all MATLAB codes that are used to carry out the experiments has been given in the previous chapter, in this one chapter first of all results of the previous thesis will be analyzed in order to detect problems and plan an strategy to improve this given algorithm. As final results there is going to propose an improved version of the older versions

First of all experiments that are used to check if the previous thesis's code and results are going on well, the thesis wrote by Sina Ghassemi (10). As next step experiments and results obtained with this code will be explained, with plenty of commentaries and conclusions.

Before starting those next subchapters, there premises and conditions that are assumed in (10) should be explained. He assumes that all videos with no Motion JPEG, which implies that there are going to be compressed frames and GOPs, the GOPs are of a fixed size. In next experiments this will be a key to make a noticeable improvement, because we will see that this does not happen, for example, in some *Apple* devices. Otherwise, another important assumption that maybe is wrong is that all videos have as first frame an I-frame. Although it is the most normal case, there is going to be seen that it is not always happened.

So let's start with the explanations of the experiments are carried on. When there are mentioned 5 methods there are referred to: first I-frame (first method), first 10 I-frames (second method), first 10 frames (third method), first 10 I and P-frames (fourth method), and the first 10 I and P-frames but making the "weighted" measurement (fifth method). But before starting the explanation of the following experiments, in the next table are going to be described the main characteristics of cameras and provided videos:

Video	Camera	Resolution	Number of frames
Video1	Apple-iPhone4s_Azzurra	1920 x 1080	297
Video2	Apple-iPhone4s_Azzurra	1920 x 1080	310
Video3	Apple-iPhone4s_Azzurra	1920 x 1080	304
Video4	Apple-iPhone4s_Azzurra	1920 x 1080	308
Video5	Apple-iPhone4s_Azzurra	1920 x 1080	299
Video6	Apple-iPhone4s_Azzurra	1920 x 1080	298
Video7	Apple-iPhone4s_Azzurra	1920 x 1080	320
Video8	Apple_iPad3_Giulia	1920 x 1080	320
Video9	Apple_iPad3_Giulia	1920 x 1080	320
Video10	Apple_iPad3_Giulia	1920 x 1080	320
Video11	Apple_iPad3_Giulia	1920 x 1080	375
Video12	Apple_iPad3_Giulia	1920 x 1080	287
Video13	Apple_iPad3_Giulia	1920 x 1080	317
Video14	Apple_iPad3_Giulia	1920 x 1080	293
Video15	Apple_iPad3_Giulia	1920 x 1080	310
Video16	Apple_iPad3_Giulia	1920 x 1080	331
Video17	Apple_iPad3_Giulia	1920 x 1080	294
Video18	Apple_iPadAir2_Tiziano	1920 x 1080	40726
Video19	Apple_iPadAir2_Tiziano	1920 x 1080	937
Video20	Apple_iPadAir2_Tiziano	1920 x 1080	939
Video21	Apple_iPadAir2_Tiziano	1920 x 1080	913
Video22	Apple_iPadAir2_Tiziano	1920 x 1080	1023
Video23	Apple_iPadAir2_Tiziano	1920 x 1080	947
Video24	Fujifilm_FinePixS2950_Giulia	1280 x 720	421
Video25	Fujifilm_FinePixS2950_Giulia	1280 x 720	421
Video26	Fujifilm_FinePixS2950_Giulia	1280 x 720	541
Video27	Fujifilm_FinePixS2950_Giulia	1280 x 720	451
Video28	Fujifilm_FinePixS2950_Giulia	1280 x 720	781
Video29	Fujifilm_FinePixS2950_Giulia	1280 x 720	511
Video30	Fujifilm_FinePixS2950_Giulia	1280 x 720	451
Video31	Fujifilm_FinePixS2950_Giulia	1280 x 720	451
Video32	Fujifilm_FinePixS2950_Giulia	1280 x 720	601
Video33	Fujifilm_FinePixS2950_Giulia	1280 x 720	601
Video34	Fujifilm_FinePixS2950_Giulia	1280 x 720	631
Video35	Fujifilm_FinePixS2950_Giulia	1280 x 720	691
Video36	LG_Nexus4_Giulio	1920 x 1080	1443
Video37	LG_Nexus4_Giulio	1920 x 1080	1614
Video38	LG_Nexus4_Giulio	1920 x 1080	146
Video39	LG_Nexus4_Giulio	1920 x 1080	502
Video40	LG_Nexus4_Giulio	1920 x 1080	998
Video41	LG_Nexus4_Giulio	1920 x 1080	192

Video42	LG_Nexus4_Giulio	1920 x 1080	1785
Video43	LG_Nexus4_Giulio	1920 x 1080	204
Video44	LG_Nexus4_Giulio	1920 x 1080	262
Video45	Nikon_D3100_Diego	1920 x 1080	2530
Video46	Nikon_D3100_Diego	1920 x 1080	499
Video47	Nikon_D3100_Diego	1920 x 1080	2380
Video48	Nikon_D3100_Diego	1920 x 1080	904
Video49	Nikon_D3100_Diego	1920 x 1080	676
Video50	Nikon_CoolpixS3000_Tiziano	640 x 480	917
Video51	Nikon_CoolpixS3000_Tiziano	640 x 480	1001
Video52	Nikon_CoolpixS3000_Tiziano	640 x 480	194
Video53	Nikon_CoolpixS3000_Tiziano	640 x 480	595
Video54	Nikon_CoolpixS3000_Tiziano	640 x 480	561
Video55	Olympus_C5500_Tomas	320 x 240	348
Video56	Olympus_C5500_Tomas	320 x 240	378
Video57	Olympus_C5500_Tomas	320 x 240	408
Video58	Olympus_C5500_Tomas	320 x 240	444
Video59	Olympus_C5500_Tomas	320 x 240	330
Video60	Olympus_C5500_Tomas	320 x 240	324
Video61	Olympus_C5500_Tomas	320 x 240	324
Video62	Olympus_C5500_Tomas	320 x 240	324
Video63	Olympus_C5500_Tomas	320 x 240	372
Video64	Olympus_C5500_Tomas	320 x 240	181
Video65	Olympus_FE5035_Tomas	640 x 480	492
Video66	Olympus_FE5035_Tomas	640 x 480	330
Video67	Olympus_FE5035_Tomas	640 x 480	348
Video68	Olympus_FE5035_Tomas	640 x 480	432
Video69	Olympus_FE5035_Tomas	640 x 480	348
Video70	Olympus_FE5035_Tomas	640 x 480	354
Video71	Olympus_FE5035_Tomas	640 x 480	378
Video72	Olympus_FE5035_Tomas	640 x 480	558
Video73	Olympus_FE5035_Tomas	640 x 480	318
Video74	Olympus_FE5035_Tomas	640 x 480	456
Video75	Olympus_FE5035_Tomas	640 x 480	372
Video76	Olympus_FE5035_Tomas	640 x 480	378
Video77	Panasonic_DMC_LZ2_Chiara	320 x 480	316
Video78	Panasonic_DMC_LZ2_Chiara	320 x 480	271
Video79	Panasonic_DMC_LZ2_Chiara	320 x 480	301
Video80	Panasonic_DMC_LZ2_Chiara	320 x 480	301
Video81	Panasonic_DMC_LZ2_Chiara	320 x 480	316
Video82	Panasonic_DMC_LZ2_Chiara	320 x 480	271
Video83	Panasonic_DMC_LZ2_Chiara	320 x 480	301
Video84	Panasonic_DMC_LZ2_Chiara	320 x 480	301

Video85	Panasonic_DMC_LZ2_Chiara	320 x 480	301
Video86	Panasonic_DMC_LZ2_Chiara	320 x 480	316
Video87	Samsung_Galaxy_Nexus_Ale	1280 x 720	489
Video88	Samsung_Galaxy_Nexus_Ale	1280 x 720	3579
Video89	Samsung_Galaxy_Nexus_Ale	1280 x 720	277
Video90	Samsung_Galaxy_Nexus_Ale	1280 x 720	1117
Video91	Samsung_Galaxy_Nexus_Ale	1280 x 720	2624
Video92	Samsung_Galaxy_Nexus_Ale	1280 x 720	248
Video93	Samsung_Galaxy_Nexus_Ale	1280 x 720	246
Video94	Samsung_Galaxy_Nexus_Ale	1280 x 720	435
Video95	Samsung_ES60_Sophie	640 x 480	296
Video96	Samsung_ES60_Sophie	640 x 480	319
Video97	Samsung_ES60_Sophie	640 x 480	296
Video98	Samsung_ES60_Sophie	640 x 480	362
Video99	Samsung_ES60_Sophie	640 x 480	295
Video100	Samsung_ES60_Sophie	640 x 480	293
Video101	Samsung_ES60_Sophie	640 x 480	292
Video102	Samsung_ES60_Sophie	640 x 480	315
Video103	Samsung_ES60_Sophie	640 x 480	300
Video104	Samsung_ES60_Sophie	640 x 480	313
Video105	Samsung_ES60_Sophie	640 x 480	338
Video106	Samsung_GT-I9070P_Tiziano	1280 x 720	6309

Table 4.1. Characteristics of cameras, videos taken and their length measured in frames.

4.1) Experiment 1: Obtaining GOP structures for start running code.

To make the MATLAB code run correctly first of all we need to know which the GOP size for each is given cameras for experiments. That is because in both of the provided MATLAB scripts one called *VIDEO_FingerprintEstimation_Sina.m* and the other called *VIDEO_RefFingerprintEstimation_Sina.m* that are previously explained in chapter 3, both use the function *GetFingerprint_video_arrayEdited* which as an input parameter it needs the GOP size of the videos that are being analyzed to estimate their Fingerprint. So, for detecting those GOP sizes there are used some software like *Qualify, Zond* and *GSpot*. But the most useful software will be the tool called *MediaInfo*, which is provided of graphical interface to better understand its use. This mainly works having as an input a certain video, and as output the software brings back information of several parameters of that video. In the next table those sizes are illustrated with a brief scheme of GOP structures for each camera. Notice that "M" is the frame separation between I and P frames, and "N" is the full GOP size (frame separation between I frames).

CAMERA NAME	GOP STRUCTURE	GOP SIZE
Apple iPad 3	IPPPPPP…PPPI… (30 I-I)	M=1, N=30
Apple iPad Air 2	IPPPPPP…PPPI… (30 I-I)	M=1, N=30
Apple iPhone 4S	IPPPPPP…PPPI… (30 I-I)	M=1, N=30
Fujifilm Fine Pix S2950	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
LG Nexus 4	IPPPPPP…PPPI… (31 I-I)	M=1, N=31
Nikon Coolpix S3000	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
Nikon D3100	IBBPBBPBB…PBBI… (12 I-I)	M=3, N=12
Olympus C5500	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
Olympus FE5035	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
Panasonic DMC LZ2	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
Samsung Galaxy Nexus Ale	IPPPPPP…PPPI… (29 I-I)	M=1, N=29
Samsung ES60	IIIIIII…	M=1, N=1 (No GOP, MJPEG)
Samsung GTI9070P	IPPPPPP…PPPI… (31 I-I)	M=1, N=31

Table 4.1.1. Information of GOPs obtained by using MediaInfo software.

4.2) Experiment 2: Checking input sequences of frames for first two methods for all 106 videos.

For this experiment several steps are followed to achieve its main objective: make sure that correct frames are taken for first two methods for estimation of fingerprints. The main reason is that for the first two methods some certain frames have to be chosen (first I-frame for the first method and first 10-frames for the second), so first of all those two methods will be chosen for checking whether their input frame sequence is the correct one.

So, first of all variables that contain the frame structures for each video are going to be calculated using *VIDEO_GetFrameInfo_File_and_Array.m* MATLAB script, where there are used the two functions that the first of them called *Get_Frame_Info* (calculates the output information text files for each video) and *Get_Frame_Array* (calculates from those previously estimated text files cell-arrays that contain the frame structure of the videos).

There is concluded that the most of the bad results of Sina's code are come from bad estimations of Fingerprints. There is observed that for cameras with Motion JPEG (MJPEG) the estimations are good because all frames are I-frames so there is no probe that frames are well taken in those cameras. So for cameras that do not have MJPEG probably the estimations for both Reference and Natural are not well done.

So for that, MATLAB scripts called *VIDEO_CheckFrames_FirstMethod.m* and *VIDEO_CheckFrames_SecondMethod.m* are used to check in all cameras if correct frames are taken for estimation and mainly it is more interesting to see results in cameras without Motion JPEG.

In the next tables results of both output text information file results are illustrated.

Video	Camera	Sequence
Video1	Apple-iPhone4s_Azzurra	I
Video2	Apple-iPhone4s_Azzurra	I
Video3	Apple-iPhone4s_Azzurra	I
Video4	Apple-iPhone4s_Azzurra	I
Video5	Apple-iPhone4s_Azzurra	I
Video6	Apple-iPhone4s_Azzurra	I
Video7	Apple-iPhone4s_Azzurra	I
Video8	Apple_iPad3_Giulia	I
Video9	Apple_iPad3_Giulia	I
Video10	Apple_iPad3_Giulia	I
Video11	Apple_iPad3_Giulia	I
Video12	Apple_iPad3_Giulia	I
Video13	Apple_iPad3_Giulia	I
Video14	Apple_iPad3_Giulia	I
Video15	Apple_iPad3_Giulia	I
Video16	Apple_iPad3_Giulia	I
Video17	Apple_iPad3_Giulia	I
Video18	Apple_iPadAir2_Tiziano	I
Video19	Apple_iPadAir2_Tiziano	I
Video20	Apple_iPadAir2_Tiziano	I
Video21	Apple_iPadAir2_Tiziano	I
Video22	Apple_iPadAir2_Tiziano	I
Video23	Apple_iPadAir2_Tiziano	I
Video24	Fujifilm_FinePixS2950_Giulia	I
Video25	Fujifilm_FinePixS2950_Giulia	I
Video26	Fujifilm_FinePixS2950_Giulia	I
Video27	Fujifilm_FinePixS2950_Giulia	I
Video28	Fujifilm_FinePixS2950_Giulia	I
Video29	Fujifilm_FinePixS2950_Giulia	I
Video30	Fujifilm_FinePixS2950_Giulia	I
Video31	Fujifilm_FinePixS2950_Giulia	I
Video32	Fujifilm_FinePixS2950_Giulia	I
Video33	Fujifilm_FinePixS2950_Giulia	I
Video34	Fujifilm_FinePixS2950_Giulia	I
Video35	Fujifilm_FinePixS2950_Giulia	I
Video36	LG_Nexus4_Giulio	I
Video37	LG_Nexus4_Giulio	I
Video38	LG_Nexus4_Giulio	I

Video39	LG_Nexus4_Giulio	I
Video40	LG_Nexus4_Giulio	I
Video41	LG_Nexus4_Giulio	I
Video42	LG_Nexus4_Giulio	I
Video43	LG_Nexus4_Giulio	I
Video44	LG_Nexus4_Giulio	I
Video45	Nikon_D3100_Diego	B
Video46	Nikon_D3100_Diego	B
Video47	Nikon_D3100_Diego	B
Video48	Nikon_D3100_Diego	B
Video49	Nikon_D3100_Diego	B
Video50	Nikon_CoolpixS3000_Tiziano	I
Video51	Nikon_CoolpixS3000_Tiziano	I
Video52	Nikon_CoolpixS3000_Tiziano	I
Video53	Nikon_CoolpixS3000_Tiziano	I
Video54	Nikon_CoolpixS3000_Tiziano	I
Video55	Olympus_C5500_Tomas	I
Video56	Olympus_C5500_Tomas	I
Video57	Olympus_C5500_Tomas	I
Video58	Olympus_C5500_Tomas	I
Video59	Olympus_C5500_Tomas	I
Video60	Olympus_C5500_Tomas	I
Video61	Olympus_C5500_Tomas	I
Video62	Olympus_C5500_Tomas	I
Video63	Olympus_C5500_Tomas	I
Video64	Olympus_C5500_Tomas	I
Video65	Olympus_FE5035_Tomas	I
Video66	Olympus_FE5035_Tomas	I
Video67	Olympus_FE5035_Tomas	I
Video68	Olympus_FE5035_Tomas	I
Video69	Olympus_FE5035_Tomas	I
Video70	Olympus_FE5035_Tomas	I
Video71	Olympus_FE5035_Tomas	I
Video72	Olympus_FE5035_Tomas	I
Video73	Olympus_FE5035_Tomas	I
Video74	Olympus_FE5035_Tomas	I
Video75	Olympus_FE5035_Tomas	I
Video76	Olympus_FE5035_Tomas	I
Video77	Panasonic_DMC_LZ2_Chiara	I
Video78	Panasonic_DMC_LZ2_Chiara	I
Video79	Panasonic_DMC_LZ2_Chiara	I
Video80	Panasonic_DMC_LZ2_Chiara	I
Video81	Panasonic_DMC_LZ2_Chiara	I

Video82	Panasonic_DMC_LZ2_Chiara	I
Video83	Panasonic_DMC_LZ2_Chiara	I
Video84	Panasonic_DMC_LZ2_Chiara	I
Video85	Panasonic_DMC_LZ2_Chiara	I
Video86	Panasonic_DMC_LZ2_Chiara	I
Video87	Samsung_Galaxy_Nexus_Ale	I
Video88	Samsung_Galaxy_Nexus_Ale	I
Video89	Samsung_Galaxy_Nexus_Ale	I
Video90	Samsung_Galaxy_Nexus_Ale	I
Video91	Samsung_Galaxy_Nexus_Ale	I
Video92	Samsung_Galaxy_Nexus_Ale	I
Video93	Samsung_Galaxy_Nexus_Ale	I
Video94	Samsung_Galaxy_Nexus_Ale	I
Video95	Samsung_ES60_Sophie	I
Video96	Samsung_ES60_Sophie	I
Video97	Samsung_ES60_Sophie	I
Video98	Samsung_ES60_Sophie	I
Video99	Samsung_ES60_Sophie	I
Video100	Samsung_ES60_Sophie	I
Video101	Samsung_ES60_Sophie	I
Video102	Samsung_ES60_Sophie	I
Video103	Samsung_ES60_Sophie	I
Video104	Samsung_ES60_Sophie	I
Video105	Samsung_ES60_Sophie	I
Video106	Samsung_GT-I9070P_Tiziano	I

Table 4.2.1. First method sequence table.

For the second method, notice that only 5 GOPs are taken as a value of GOPs that all videos at least have, in order to get easier the process and detect at least cameras that do not have correct input even for 5 GOPs only.

Video	**Camera**	**Sequence**
Video1	Apple-iPhone4s_Azzurra	IIIII
Video2	Apple-iPhone4s_Azzurra	IIIII
Video3	Apple-iPhone4s_Azzurra	IIIII
Video4	Apple-iPhone4s_Azzurra	IIIII
Video5	Apple-iPhone4s_Azzurra	IIIII
Video6	Apple-iPhone4s_Azzurra	IIIPP
Video7	Apple-iPhone4s_Azzurra	IIIII

Video8	Apple_iPad3_Giulia	IIIII
Video9	Apple_iPad3_Giulia	IIIII
Video10	Apple_iPad3_Giulia	IIPPP
Video11	Apple_iPad3_Giulia	IIIII
Video12	Apple_iPad3_Giulia	IIIII
Video13	Apple_iPad3_Giulia	IIIII
Video14	Apple_iPad3_Giulia	IIIII
Video15	Apple_iPad3_Giulia	IIIII
Video16	Apple_iPad3_Giulia	IIIII
Video17	Apple_iPad3_Giulia	IPPPP
Video18	Apple_iPadAir2_Tiziano	IIIII
Video19	Apple_iPadAir2_Tiziano	IIIII
Video20	Apple_iPadAir2_Tiziano	IIIII
Video21	Apple_iPadAir2_Tiziano	IIIII
Video22	Apple_iPadAir2_Tiziano	IIIII
Video23	Apple_iPadAir2_Tiziano	IIIII
Video24	Fujifilm_FinePixS2950_Giulia	IIIII
Video25	Fujifilm_FinePixS2950_Giulia	IIIII
Video26	Fujifilm_FinePixS2950_Giulia	IIIII
Video27	Fujifilm_FinePixS2950_Giulia	IIIII
Video28	Fujifilm_FinePixS2950_Giulia	IIIII
Video29	Fujifilm_FinePixS2950_Giulia	IIIII
Video30	Fujifilm_FinePixS2950_Giulia	IIIII
Video31	Fujifilm_FinePixS2950_Giulia	IIIII
Video32	Fujifilm_FinePixS2950_Giulia	IIIII
Video33	Fujifilm_FinePixS2950_Giulia	IIIII
Video34	Fujifilm_FinePixS2950_Giulia	IIIII
Video35	Fujifilm_FinePixS2950_Giulia	IIIII
Video36	LG_Nexus4_Giulio	IIIII
Video37	LG_Nexus4_Giulio	IIIII
Video38	LG_Nexus4_Giulio	IIIII
Video39	LG_Nexus4_Giulio	IIIII
Video40	LG_Nexus4_Giulio	IIIII
Video41	LG_Nexus4_Giulio	IIIII
Video42	LG_Nexus4_Giulio	IIIII
Video43	LG_Nexus4_Giulio	IIIII
Video44	LG_Nexus4_Giulio	IIIII
Video45	Nikon_D3100_Diego	BBBBB
Video46	Nikon_D3100_Diego	BBBBB
Video47	Nikon_D3100_Diego	BBBBB
Video48	Nikon_D3100_Diego	BBBBB
Video49	Nikon_D3100_Diego	BBBBB
Video50	Nikon_CoolpixS3000_Tiziano	IIIII

Video51	Nikon_CoolpixS3000_Tiziano	IIIII
Video52	Nikon_CoolpixS3000_Tiziano	IIIII
Video53	Nikon_CoolpixS3000_Tiziano	IIIII
Video54	Nikon_CoolpixS3000_Tiziano	IIIII
Video55	Olympus_C5500_Tomas	IIIII
Video56	Olympus_C5500_Tomas	IIIII
Video57	Olympus_C5500_Tomas	IIIII
Video58	Olympus_C5500_Tomas	IIIII
Video59	Olympus_C5500_Tomas	IIIII
Video60	Olympus_C5500_Tomas	IIIII
Video61	Olympus_C5500_Tomas	IIIII
Video62	Olympus_C5500_Tomas	IIIII
Video63	Olympus_C5500_Tomas	IIIII
Video64	Olympus_C5500_Tomas	IIIII
Video65	Olympus_FE5035_Tomas	IIIII
Video66	Olympus_FE5035_Tomas	IIIII
Video67	Olympus_FE5035_Tomas	IIIII
Video68	Olympus_FE5035_Tomas	IIIII
Video69	Olympus_FE5035_Tomas	IIIII
Video70	Olympus_FE5035_Tomas	IIIII
Video71	Olympus_FE5035_Tomas	IIIII
Video72	Olympus_FE5035_Tomas	IIIII
Video73	Olympus_FE5035_Tomas	IIIII
Video74	Olympus_FE5035_Tomas	IIIII
Video75	Olympus_FE5035_Tomas	IIIII
Video76	Olympus_FE5035_Tomas	IIIII
Video77	Panasonic_DMC_LZ2_Chiara	IIIII
Video78	Panasonic_DMC_LZ2_Chiara	IIIII
Video79	Panasonic_DMC_LZ2_Chiara	IIIII
Video80	Panasonic_DMC_LZ2_Chiara	IIIII
Video81	Panasonic_DMC_LZ2_Chiara	IIIII
Video82	Panasonic_DMC_LZ2_Chiara	IIIII
Video83	Panasonic_DMC_LZ2_Chiara	IIIII
Video84	Panasonic_DMC_LZ2_Chiara	IIIII
Video85	Panasonic_DMC_LZ2_Chiara	IIIII
Video86	Panasonic_DMC_LZ2_Chiara	IIIII
Video87	Samsung_Galaxy_Nexus_Ale	IIIII
Video88	Samsung_Galaxy_Nexus_Ale	IIIII
Video89	Samsung_Galaxy_Nexus_Ale	IIIII
Video90	Samsung_Galaxy_Nexus_Ale	IIIII
Video91	Samsung_Galaxy_Nexus_Ale	IIIII
Video92	Samsung_Galaxy_Nexus_Ale	IIIII
Video93	Samsung_Galaxy_Nexus_Ale	IIIII

Video94	Samsung_Galaxy_Nexus_Ale	IIIII
Video95	Samsung_ES60_Sophie	IIIII
Video96	Samsung_ES60_Sophie	IIIII
Video97	Samsung_ES60_Sophie	IIIII
Video98	Samsung_ES60_Sophie	IIIII
Video99	Samsung_ES60_Sophie	IIIII
Video100	Samsung_ES60_Sophie	IIIII
Video101	Samsung_ES60_Sophie	IIIII
Video102	Samsung_ES60_Sophie	IIIII
Video103	Samsung_ES60_Sophie	IIIII
Video104	Samsung_ES60_Sophie	IIIII
Video105	Samsung_ES60_Sophie	IIIII
Video106	Samsung_GT-I9070P_Tiziano	IIIII

Table 4.2.2. Second method table.

So as we can see attending to the first table we conclude that for the camera *Nikon_D3100_Diego* the first frame is not an I-frame like is assumed in (10), and in this table is probed that for that camera the estimation for the first method is not well done. For the second table we conclude that *Apple* devices do not use fixed GOP size, because is probed there that the increment of GOP size used to take only I-frames which it should work in fixed GOP structure does not take only I-frames so there is deduced *Apple* device videos where this happen use variable GOP size. So for analyzing that, there is also used the second script to do a further analysis in those devices to probe the previous conclusion of the *Apple* devices use variable GOP size and that this is the main reason that Sina's code does not work well in those devices. In the next table more GOPs are taken (10 GOPs) for those videos to make a better analysis:

Video	**Camera**	**Sequence**
Video1	Apple-iPhone4s_Azzurra	IIIIIIIII
Video2	Apple-iPhone4s_Azzurra	IIIIIIIII
Video3	Apple-iPhone4s_Azzurra	IIIIIIIII
Video4	Apple-iPhone4s_Azzurra	IIIIIIIII
Video5	Apple-iPhone4s_Azzurra	IIIIIIIII
Video6	Apple-iPhone4s_Azzurra	IIIPPPPPPPP
Video7	Apple-iPhone4s_Azzurra	IIIIIIIII

Video8	Apple_iPad3_Giulia	IIIIIIIIII
Video9	Apple_iPad3_Giulia	IIIIIIIIII
Video10	Apple_iPad3_Giulia	IIPPPPPPPPP
Video11	Apple_iPad3_Giulia	IIIIIIIIII
Video12	Apple_iPad3_Giulia	IIIIIIIIII
Video13	Apple_iPad3_Giulia	IIIIIIIIII
Video14	Apple_iPad3_Giulia	IIIIIIIIII
Video15	Apple_iPad3_Giulia	IIIIIIIIII
Video16	Apple_iPad3_Giulia	IIIIIIIIII
Video17	Apple_iPad3_Giulia	IPPPPPPPPPP
Video18	Apple_iPadAir2_Tiziano	IIIIIIIIII
Video19	Apple_iPadAir2_Tiziano	IIIIIIIIII
Video20	Apple_iPadAir2_Tiziano	IIIIIIIIII
Video21	Apple_iPadAir2_Tiziano	IIIIIIIIII
Video22	Apple_iPadAir2_Tiziano	IIIIIIIIII
Video23	Apple_iPadAir2_Tiziano	IIIIIIIIII

Table 4.2.3. Table for Apple device analysis.

As we can see is demonstrated that in the case of those videos that are P-frames in the input frame array the GOP size is variable because if we take from the first frame an increment of the fixed GOP size that Sina proposes in his vector of GOP sizes that are similar to those that estimated in the first experiment, all frames should be I-frames. So there it can be seen that not in all videos a fixed GOP is applied.

As conclusion for all checking method results we can conclude that Sina's approach works well in videos that there is fixed GOP size, and in the case of the most videos this happens, but for well working those videos must have as first frame an I-frame because if not same case of *Nikon_D3100_Diego* will happen.

4.3) Experiment 3: Checking input sequences of frames for last two methods for videos with no Motion JPEG.

As previously said, this analysis is applied for those cameras which do not use Motion JPEG (no GOP, all I-frames structure). First of all which frames are taken for those for this method will be analyzed, to make sure that correct frames are being taken in those cameras. For doing this other MATLAB script is done that is explained in the chapter 3, called *VIDEO_CheckFrames_Fourth_FifthMethod.m*, which as the previous frame checking scripts, it gives a output text information file with the frame structure is taken for those methods.

Otherwise, an Excel project is taken called *CorrValuesThresholding.xlsx* that contains values of computed correlation processes in the MATLAB script called *VIDEO_CorrelatedArray_Thresholding.m*, where correlation is computed between both Natural and Reference Fingerprints but with the only difference of taking as threshold for each correlation process 3 times the standard deviation, so converts the output confusion matrix into a binary one, assuming that in each value is going to be "1" when the correlation value is higher than the mentioned threshold (different for each correlation operation), and the "0" values will mean the opposite, that those values are lower than this threshold. The results are in the next table:

Video	Camera	Sequence
Video1	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video2	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video3	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video4	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video5	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video6	Apple-iPhone4s_Azzurra	IPIPIPPPPP
Video7	Apple-iPhone4s_Azzurra	IPIPIPIPIP
Video8	Apple_iPad3_Giulia	IPIPIPIPIP
Video9	Apple_iPad3_Giulia	IPIPIPIPIP
Video10	Apple_iPad3_Giulia	IPIPPPPPPP
Video11	Apple_iPad3_Giulia	IPIPIPIPIP

Video12	Apple_iPad3_Giulia	IPIPIPIPIP
Video13	Apple_iPad3_Giulia	IPIPIPIPIP
Video14	Apple_iPad3_Giulia	IPIPIPIPIP
Video15	Apple_iPad3_Giulia	IPIPIPIPIP
Video16	Apple_iPad3_Giulia	IPIPIPIPIP
Video17	Apple_iPad3_Giulia	IPPPPPPPPP
Video18	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video19	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video20	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video21	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video22	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video23	Apple_iPadAir2_Tiziano	IPIPIPIPIP
Video36	LG_Nexus4_Giulio	IPIPIPIPIP
Video37	LG_Nexus4_Giulio	IPIPIPIPIP
Video39	LG_Nexus4_Giulio	IPIPIPIPIP
Video40	LG_Nexus4_Giulio	IPIPIPIPIP
Video41	LG_Nexus4_Giulio	IPIPIPIPIP
Video42	LG_Nexus4_Giulio	IPIPIPIPIP
Video43	LG_Nexus4_Giulio	IPIPIPIPIP
Video44	LG_Nexus4_Giulio	IPIPIPIPIP
Video45	Nikon_D3100_Diego	BBBBBBBBBB
Video46	Nikon_D3100_Diego	BBBBBBBBBB
Video47	Nikon_D3100_Diego	BBBBBBBBBB
Video48	Nikon_D3100_Diego	BBBBBBBBBB
Video49	Nikon_D3100_Diego	BBBBBBBBBB
Video56	Olympus_C5500_Tomas	IIIIIIIIII
Video57	Olympus_C5500_Tomas	IIIIIIIIII
Video58	Olympus_C5500_Tomas	IIIIIIIIII
Video59	Olympus_C5500_Tomas	IIIIIIIIII
Video60	Olympus_C5500_Tomas	IIIIIIIIII
Video61	Olympus_C5500_Tomas	IIIIIIIIII
Video62	Olympus_C5500_Tomas	IIIIIIIIII
Video63	Olympus_C5500_Tomas	IIIIIIIIII
Video64	Olympus_C5500_Tomas	IIIIIIIIII
Video77	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video78	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video79	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video80	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video81	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video82	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video83	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video84	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII
Video85	Panasonic_DMC_LZ2_Chiara	IIIIIIIIII

Video86	Panasonic_DMC_LZ2_Chiara	IIIIIIIII
Video87	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video88	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video89	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video90	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video91	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video92	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video93	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video94	Samsung_Galaxy_Nexus_Ale	IPIPIPIPIP
Video106	Samsung_GT-I9070P_Tiziano	IPIPIPIPIP

Table 4.3.1. Input frame sequences for methods 4 and 5.

There are clear in this table some new things. On one hand, we see that in the same *Apple* videos where there is variable GOP does not take the correct sequence of frames (IPIPIPIPIP…), and effectively if we see binary matrix of the Excel project for those videos in the last two methods we see null values. So these can be the reason for those low values for those videos. For the rest of *Apple* videos we can see that the correct sentence of frames is taken, so the low correlation values probably come from the Reference Fingerprint. As told before, in those devices probably the Reference Fingerprint is not well estimated so this brings us a very low correlation values back. On the other hand *Nikon_D3100_Diego* takes a wrong sentence, but correlation values are higher in most videos of this camera than in the rest. This maybe comes from that in this camera the Reference Fingerprint is badly estimated taking other frames that are not I-frames, but probably this Reference Fingerprint is so similar to the estimated Natural Fingerprints in those videos that they return high correlation values.

In cameras like *Olympus_C5500_Tomas* and *Panasonic_DMC_LZ2_Chiara* a wrong sentence is taken too, but two different cases are distinguished for both cameras. Videos in first camera have very low correlation values, but probably it happens because it has a very low image quality so the Fingerprint is not well estimated, even with a wrong sentence that has all I-frames. For the second camera we see that have higher values of correlation, and it is because the wrong sentence has all I-frames so it implies high correlation value.

4.4) Experiment 4: Repeat all experiments with well estimated Natural and Reference Fingerprints.

After detecting the main problems in Sina's different pieces of code in previous experiments that are done, those experiments are done in his thesis like obtaining the confusion matrices and ROC curves will be repeated with a new proposition of those MATLAB codes. The main problem is that correct frames are not taken for estimation for both Reference and Natural Fingerprints, so with the variables that contain the GOP structure and frame types that are generated in previous experiments are going to be used so correct frames will be chosen for the estimation. This estimation process will be the same but changing the input arrays, so the main objective of this last experiment is to see how improves results in (10) if it does.

4.4.1) Reference and Natural Fingerprint Estimation:

For the estimation of Reference Fingerprints the process will be so similar, and this mentioned process is carried on in the MATLAB script called *VIDEO_GetRefFingerprints_CorrectFrames.m* where the only change is that frame arrays of smooth videos for each camera calculated using previous functions designed to calculate those are loaded, and because 20 I-frames will be chosen for the estimation, all I-frames are found in this array saving in an index array al index numbers of those I-frames in the video. Only 20 first indexes are saved in this array, and putted as input in the Fingerprint estimator function. The GOP sizes are used are those that Sina uses in his code. This MATLAB script will calculate all of them automatically as explained in the third chapter, and later those variables will be saved in the *ReferenceFingerprintCorrect* folder in *Fingerprints* folder.

A similar function is in the *VIDEO_GetFingerprints_CorrectFrames.m* where Natural Fingerprints are estimated by using correct frames. There are 5 different ways, corresponding to the 5 different methods that are used in Sina's code. For all methods there are loaded the variables that contain the GOP structures of videos. For the first method only first I frame's index is saved in the input array, and for the second all I frames are found and only indexes of the first 10 are saved in the input array. For the third method the first 10 frame's indexes are chosen and saved in the input array, and for the last two methods two findings are done, for I-frames and for P-frames, and alternating their indexes only the first 10 indexes are chosen for each index array as input array. For those two last methods, only some videos are used, not all of them. The videos that are used are those who have P-frames, because it has no sense applying those methods in videos that only have I-frames.

4.4.2) Obtained ROC Curves and its definition:

In the next lines all obtained ROC curves are going to be explained and commented and in each one some conclusions will be deduced. Those ROC curves will be calculated from the correlation values between all Reference Fingerprints and Natural Fingerprints of the different five methods. As a theoretical basis of ROC curves and specially applied to this term I have to say it shows the relation between Detection Rate (a video that belongs to the camera has been correctly detected) and the False Alarm Rate (a video that belongs to another camera has not been detected), for each threshold value (this threshold value will be the one that we use to decide if all videos that have higher correlation value than that threshold will belong to the camera, and the lower values not). This threshold value takes all of the possible values that it can have (between 0 and 1). So "ideally" a perfect detection has a ROC curve one that always has a Detection Rate of 1 for all threshold value, so the main objective would be to design an algorithm that returns results which for all methods brings curves that have as much values of 1 as possible for all threshold values.

a) Apple-iPhone4s_Azzurra:

For this camera with IPPPPP… GOP structure we can see that has a very good detection in general for all methods. The worst one will be the second method (taking first 10 I-frames) and the best one the weighted measure of Fingerprints taking 10 I and P-frames, the fifth method. For this camera we can see the high influence of P-frames, the methods which have mixed input between I and P-frames have higher Detection Rate. The main reason for this that I-frames have no compression, so those frames have more information about the PRNU Fingerprint. But it does not explain why methods with only I-frames have worse detection. This maybe will be because the P-frames are chosen for the last two methods are from the same GOP, so between them have more correlation than with others, just because all of them have same I-frame as reference frame for the time-prediction in compression of video. For those strange results further analysis will be in a next experiment unique for *Apple* devices.

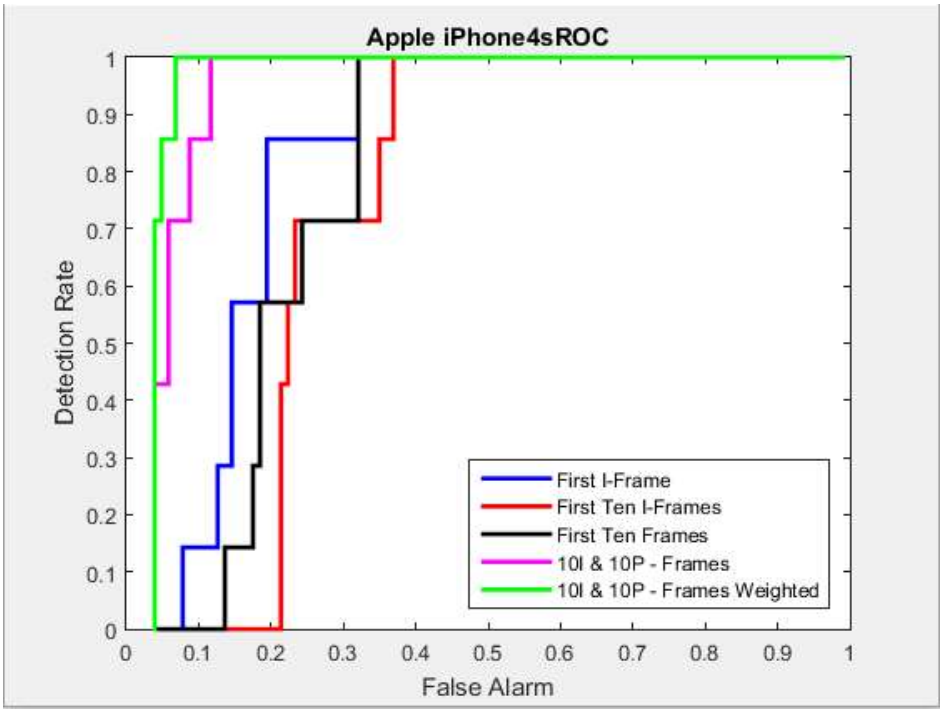

Image 4.4.2.1. ROC Curve for camera identification in Apple iPhone 4s.

b) Apple_iPad3_Giulia:

For this camera there are some other things to comment. As best method we see that it is the fifth one, and with it the first one being this a little bit worse. The fourth method seems to be quite good detection method for this camera, even not having a Detection Rate of 1 practically. For those two last methods we can conclude the same conclusion of the previous camera, because of P-frames chosen for those methods are from the same GOP. As worst method we have the third method, and this is because in the case of this camera with GOP frame structure of IPPPPP… we take as input one I-frame and 9 P-frames. So there we see the high effect of taking I-frames for detection because of their null compression.

As conclusion there is that for this camera we have good detection for the last two methods and for the first. Here we can see the importance of taking I-frames for camera identification.

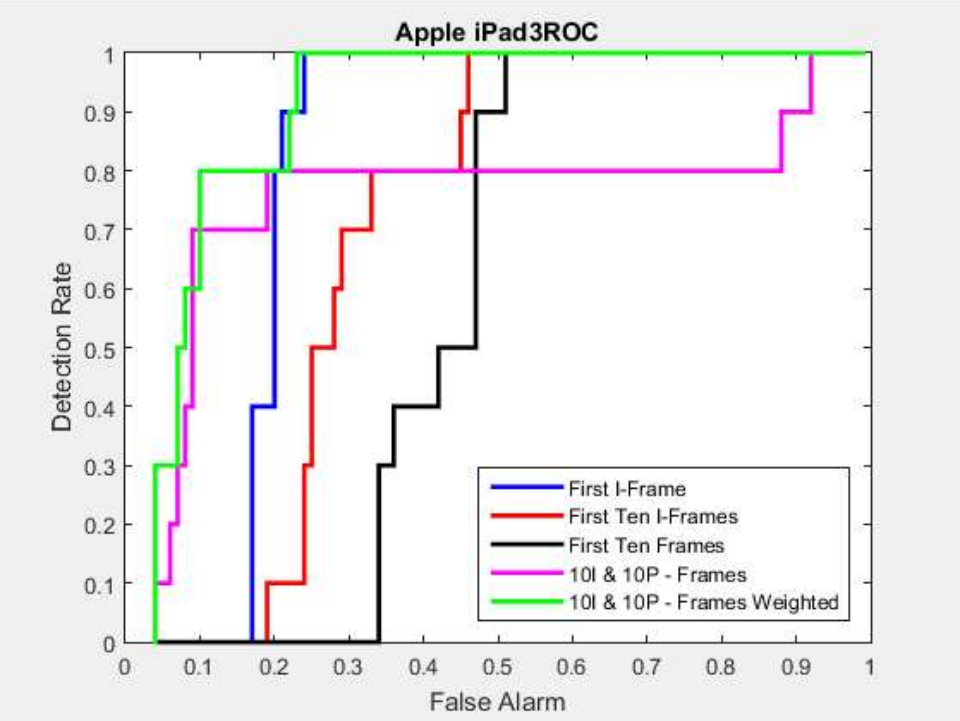

Image 4.4.2.2. ROC Curve for camera identification in Apple iPad3.

c) *Apple_iPadAir2_Tiziano:*

For this camera with IPPPP… GOP structure there can be seen a not good detection. As best method we can take the fifth method and similar to this the fourth one, because of the previous camera's conclusions (P-frames of the same GOP). For the rest of the methods we can see that the best ones are the first and second method, and it is because they only take as input for Fingerprint estimation only one I-frame and ten I-frames, respectively.

For the moment, for the *Apple* devices we conclude fifth and fourth method work well, the first method and second work in the good the way because of the only presence of I-frames, and the worst method will be the third one because of the lack of presence of those mentioned I-frames. Even so, *Apple* devices will be better analyzed in a next experiment to try to understand those strange results.

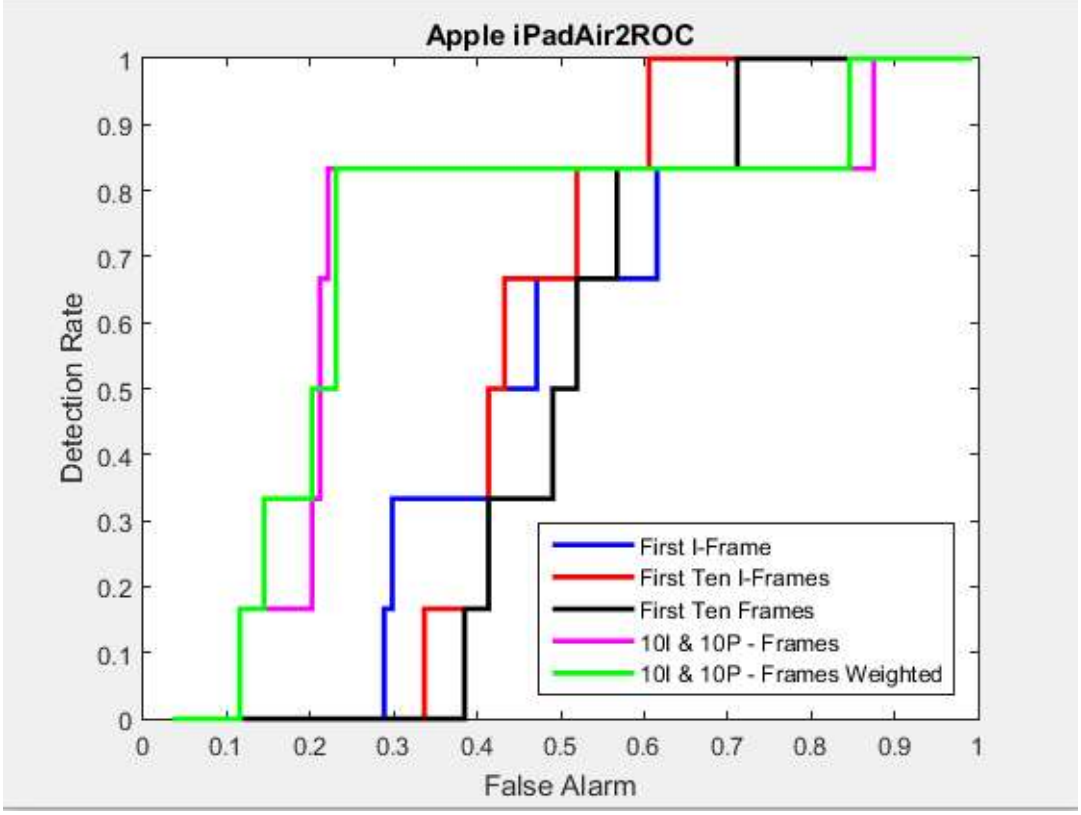

Image 4.4.2.3. ROC Curve for camera identification in Apple iPad Air2.

d) Fujifilm_FinePixS2950_Giulia:

For this camera with IIIII… frame structure (no GOP, only I-frames) and for the next cameras with a similar frame structure, the last two methods will not be evaluated in those cameras. The reason is that for those methods 10 P-frames are needed to carry on those estimations, and those cameras do not have any of those frames, so those estimations are not able to be done on those videos with no GOP and all I-frames (Motion JPEG).

So we see that the three methods are implemented on this camera are very good detectors in this case. Though, we see a very slight difference between the first method and the next two. For the first we only take one I-frame, and in the next two methods we have the same input because in this frame structure of al I-frames the first 10 frames are 10 I-frames, so the input is the same. So the slight difference between the first and the rest of the methods is because in the first we have much less I-frames, and in the other two we have more information for PRNU Fingerprint Estimation.

So as conclusion here we have that we see that is better to take as much I-frames as possible, because there is more information of the PRNU noise for a good camera detection.

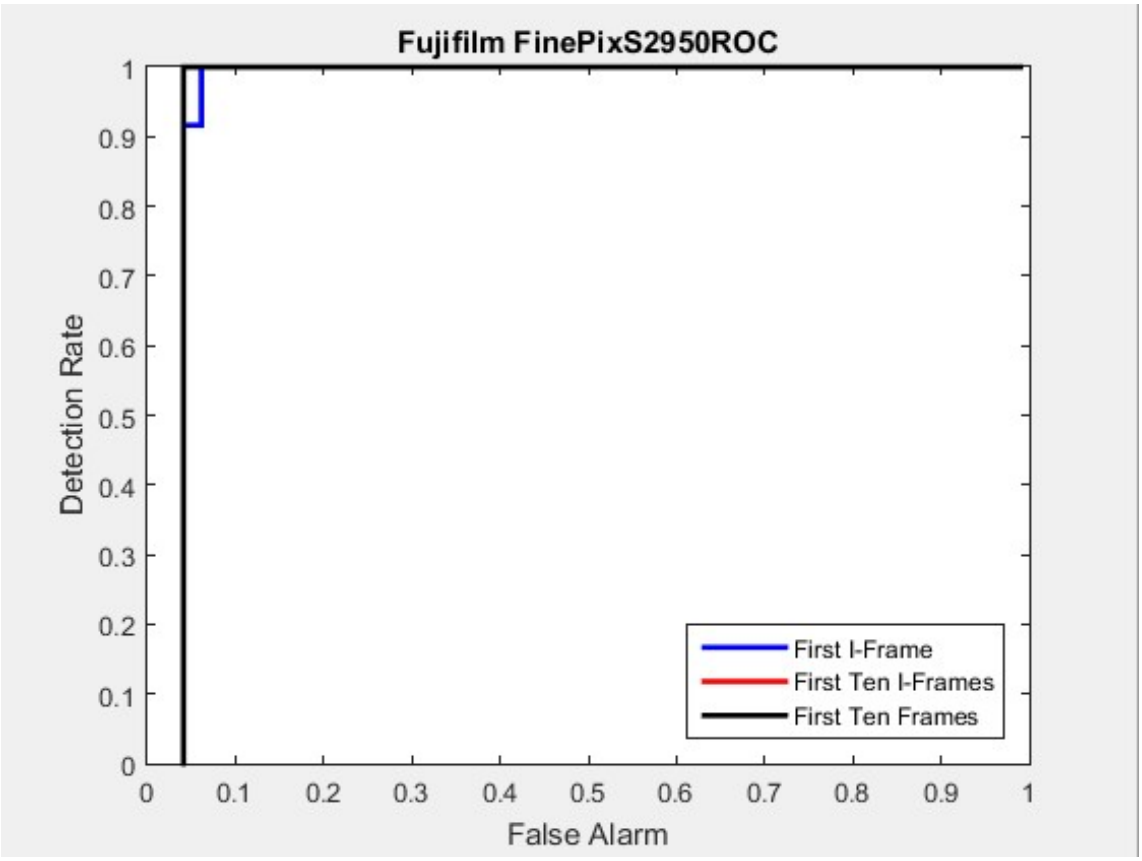

Image 4.4.2.4. ROC Curve for camera identification in Fujifilm FinePixS2950.

e) LG_Nexus4_Giulio:

In general for this camera with a GOP structure of IPPPP… we see that all methods work very well in detection. The last two methods and the second are the best ones, because those are the methods that most I-frames are taken as input for Fingerprint Estimation. We see that the first and third methods work well in detection as well, but we can consider that the first method is better than third one. That is because the third method has more P-frames than the first, and those P frames having compression on themselves have a loss of information of PRNU pattern noise.

As conclusion we have that more or less all detection methods work well in this camera, and we can clearly see again the importance of having as much I-frames as possible.

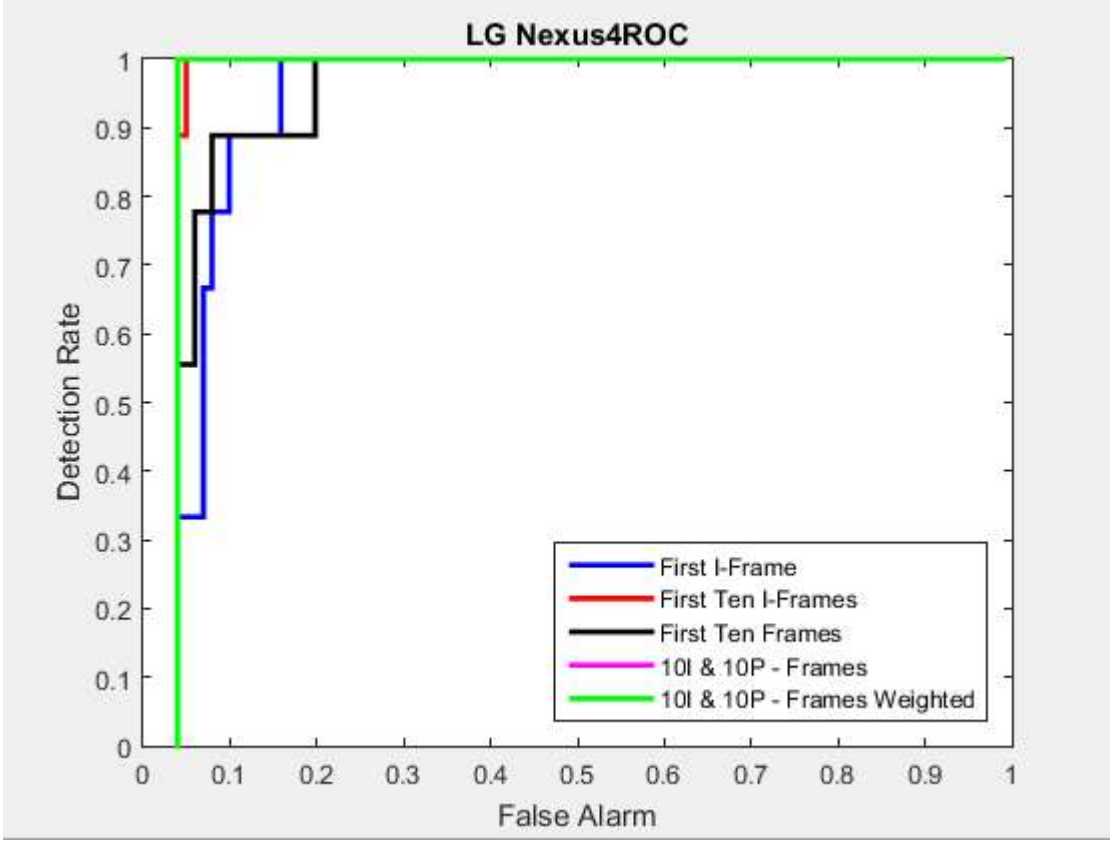

Image 4.4.2.5. ROC Curve for camera identification in LG Nexus 4.

f) Nikon_D3100_Diego:

This camera is more different from the rest cameras. It has a M=12, N=3 GOP structure, which means that the frame organization here is IBBPBBPBB…, so in the case of this camera is very important to choose correctly the frames for this method. From this MATLAB code made with choosing the correct frames, the best improvement comparing with results in (10) is in the case of this camera. Analyzing the frames taken there for estimation in previous experiments, we noticed that the first frame was not an I-frame, it was a B-frame. So making a comparison we see this detection for all methods is almost ideal in this camera choosing well the frames of the video. In the case of other videos we do not notice the improvements so much but here we can see the high importance of choosing the correct frames, and the biggest improvement of this code from Sina's is the case of this camera that has a complex GOP structure and the results taking correct frames are very good.

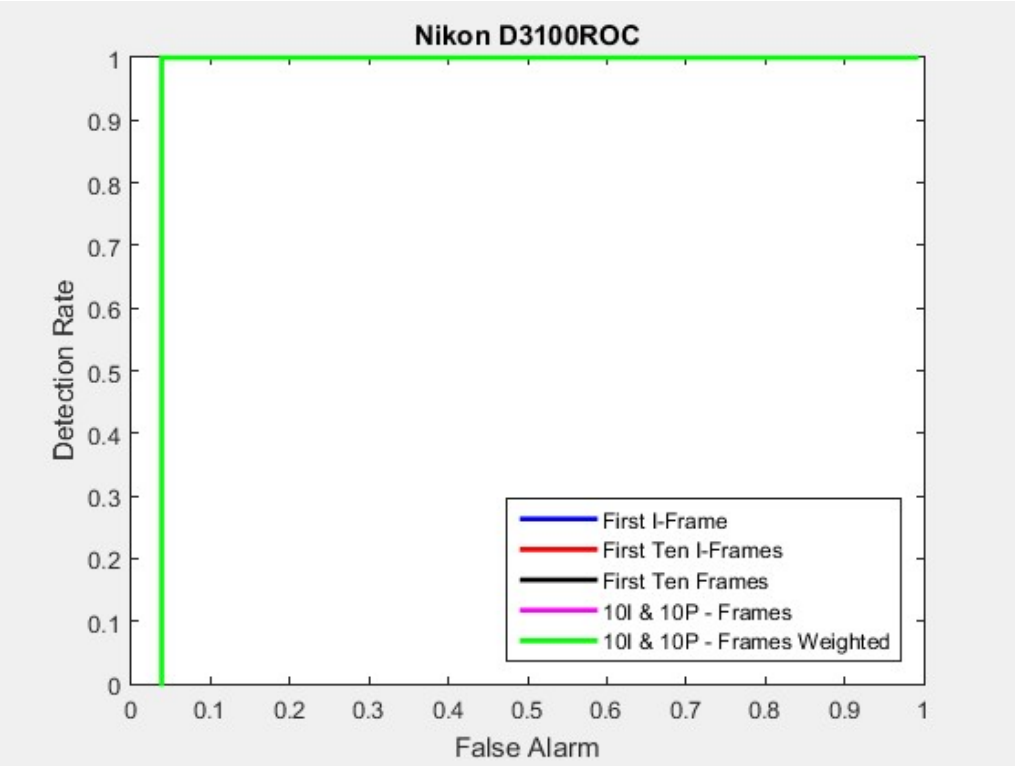

Image 4.4.2.6. ROC Curve for camera identification in Nikon D3100.

g) Nikon_CoolpixS3000_Tiziano:

For this camera with no GOP structure, just all I-frames, we see again the high importance of taking I-frames, like in the previous case of a camera with the same frame organization. The second and third method are the same (10 first frames are all I-frames), so those method have as input arrays that contain much more information than taking just one I-frame, and this difference is noticeable in the next ROC curve, with high Detection Rate for second and third methods, and quite high for the first one.

So as conclusions we have the importance of having as much frames with no compression that contain all information about PRNU noise as possible (I-frames), to make a good camera detection.

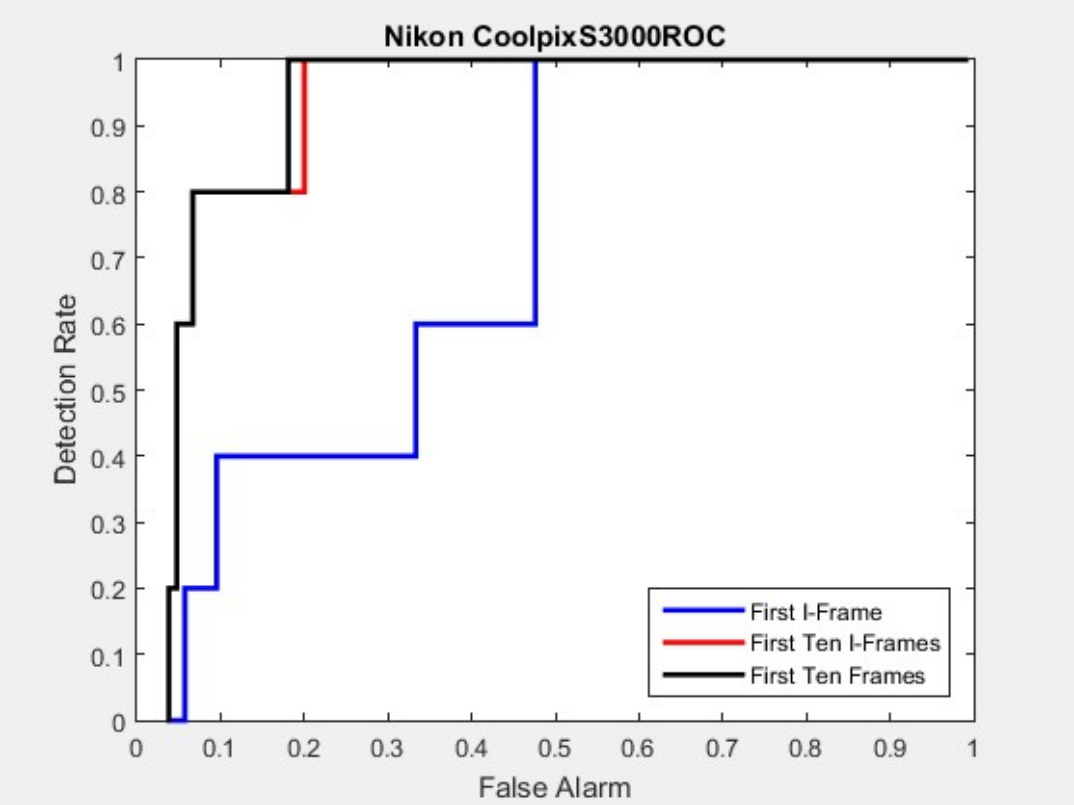

Image.4.4.2.7. ROC Curve for camera identification in Nikon Coolpix S3000.

h) Olympus_C5500_Tomas:

Here we see that the detection is not so good in this camera. Here we see that the first method in general is slightly better than the next two methods, or at least similar. For this camera we know that a very ow resolution is applied, so the reason of so low detection is because in a camera with so bad image quality Fingerprint Estimation is not as good as in a high quality video. We can see that even in a single I-frame we have slightly better detection than in 10 I-frames of so low resolution video, for both Reference and Natural videos. For this reason the correlation values and detection must be bad. So here we have another conclusion, different from the rest of cameras. Here we can see the effect of having good or bad resolution, because in the case of this camera we have bad detection because of the low quality. The higher is the image quality, the better will be the detection because Reference and Natural Fingerprints are better estimated (more information can be extracted).

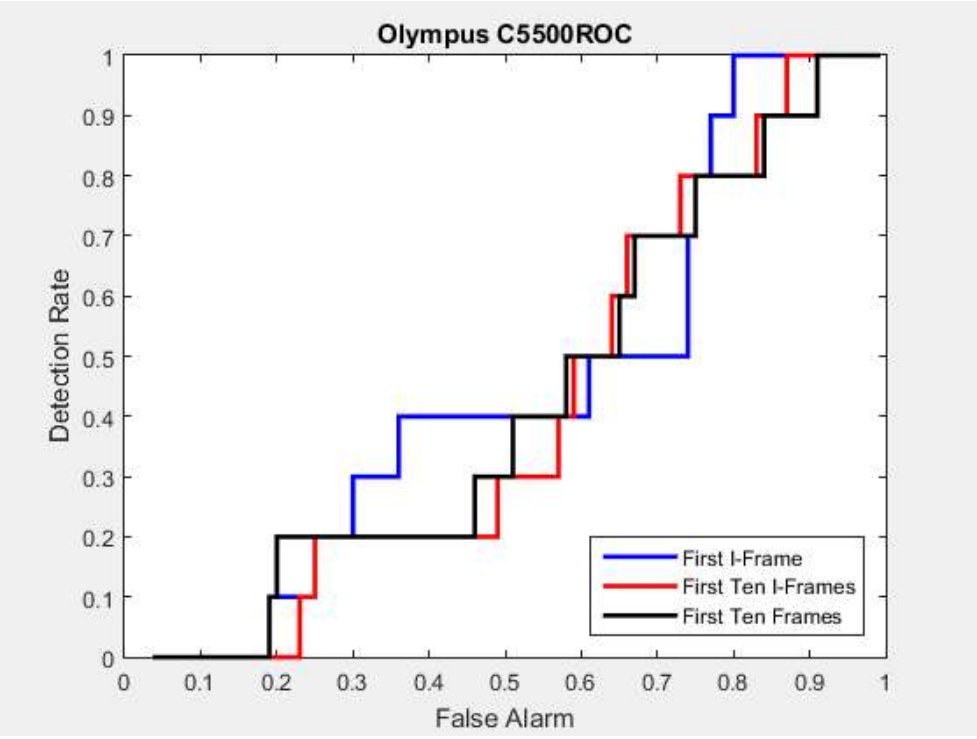

Image 4.4.2.8. ROC Curve for camera identification in Olympus C5500.

i) Olympus_FE5035_Tomas:

This camera has no GOP, all I-frames compose the frame organization in the video. So here neither last two methods will be analyzed. Here we have a camera with better quality, and we can notice it in the next ROC curve graphic. In this camera we see again another case of the importance of taking many I-frames. Here again happens that the second and third methods (same input frame structures in this video, 10 I-frames) have better detection than taking just one I-frame, the first method.

The main conclusion here is the same that we have in other high quality cameras with no GOP, and it is that taking as much I-frames as possible is better to have a good camera detection, because this carries a good Fingerprint Estimation, both Reference and Natural.

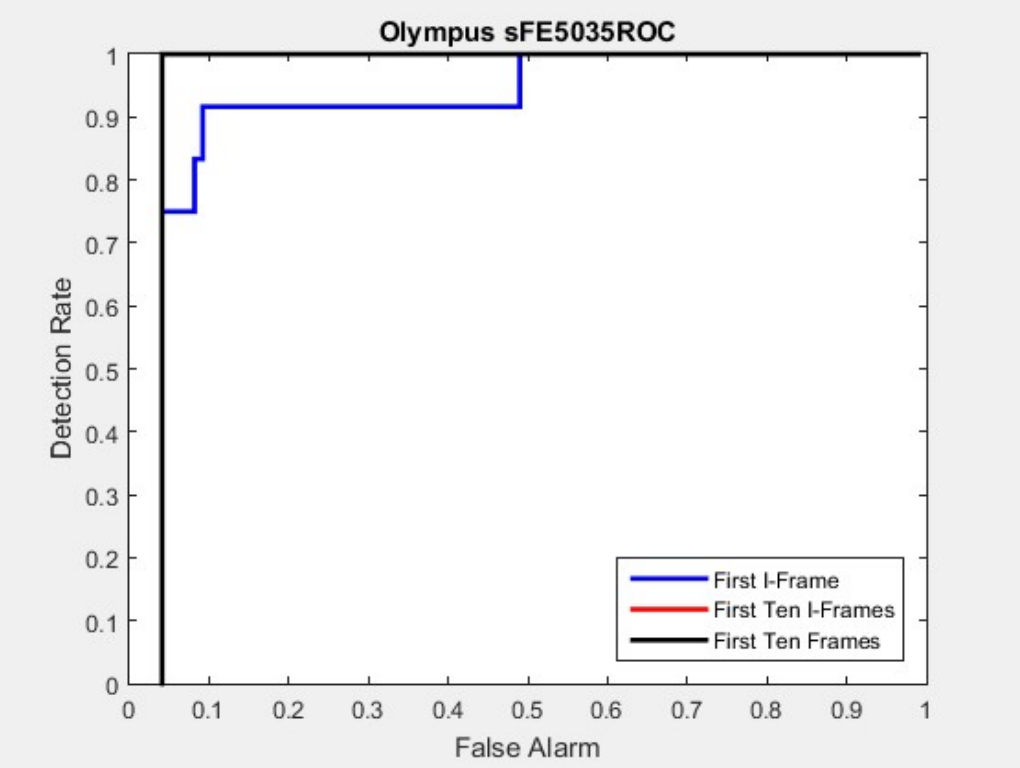

Image 4.4.2.9. ROC Curve for camera identification in Olympus sFE5035.

j) Panasonic_DMC_LZ2_Chiara:

In this camera we have the same situation of the previous camera, and it is that we have no GOP here and more or less quite higher image quality. So in this camera there is not so much to conclude or comment, because of here the interpretation is practically the same or almost the same. We again have better detections for the second and third methods (same frame array input for estimation of Natural Fingerprint, 10 I-frames) than in the first one.

So as conclusions for this camera there are the same in the previous camera analysis, and it is that having many I-frames correctly is much better than having just one, but we can notice the also good detection for the first method.

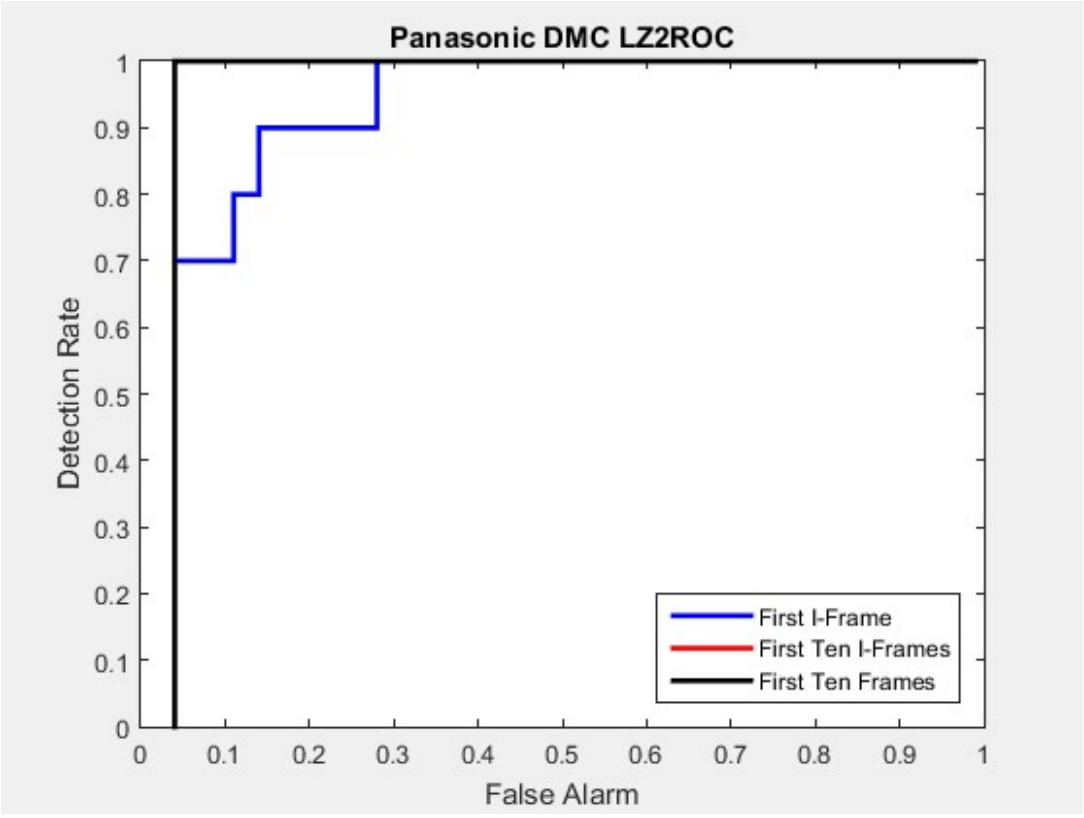

Image 4.4.2.10. ROC Curves for camera identification in Panasonic DMC LZ2.

k) Samsung_Galaxy_Nexus_Ale:

In this camera with a GOP structure of IPPPP… we can implement all estimation procedures. As general reading, we can notice that there is not so good detection for this camera. For the first 10 frames method (third one) we see that there is the best Detection Rate comparing with the other methods. The next best method would be the first one, which takes only the first I-frame, and the next one would be the second method (10 I-frames). The last two methods are the less reliable methods for detection.

This probably comes from that some of the natural videos of this camera are dark taken, so maybe this detection is affected by this low brightness in Fingerprint Estimation.

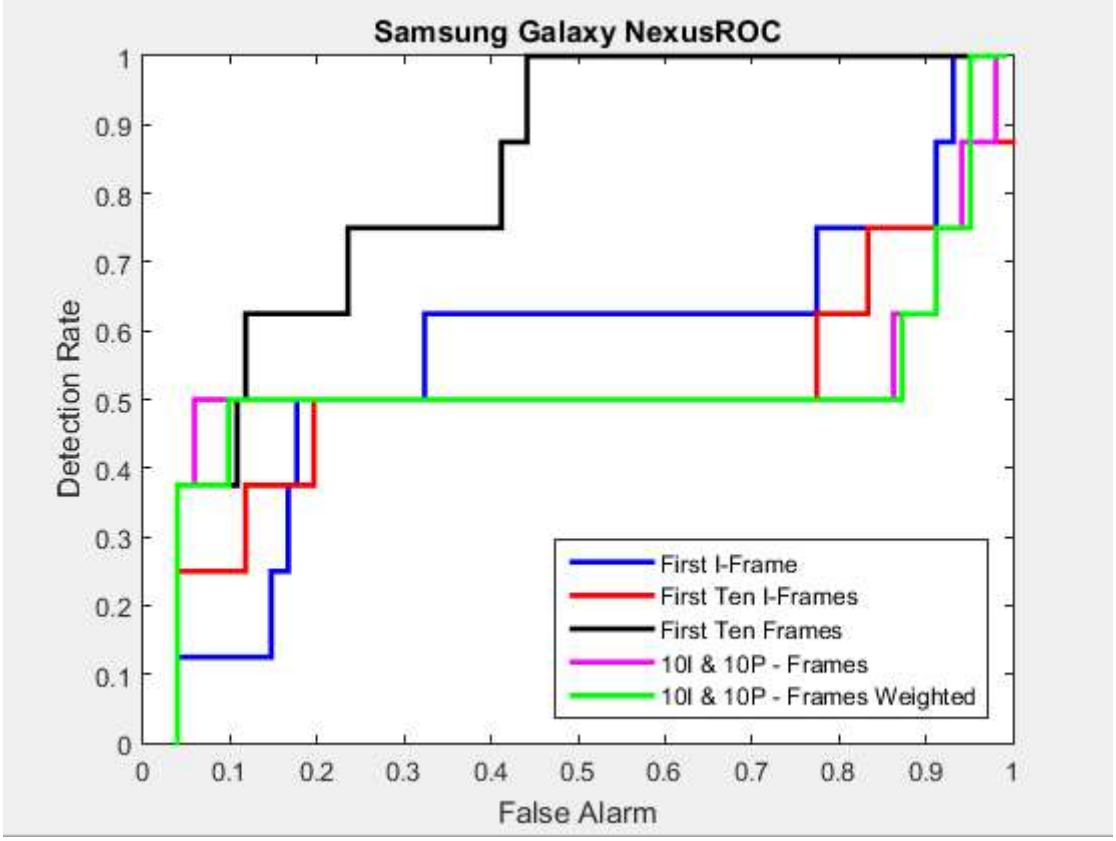

Image 4.4.2.11. ROC Curve for camera identification in Samsung Galaxy Nexus.

l) Samsung_ES60_Sophie:

For this camera we have the same situation of other cameras of normal or high resolution with no GOP and for that all I-frames, but in this case we see that the detection for the first method is as well as the next two methods, and three of them are very good methods to detect. In this case we see very well that ideally the PRNU pattern noise does remain to stay the same in all frames, and in this camera it happens, because taking the first frame or taking the first 10 returns the same results.

So for this, the conclusion we have is that the detection is very good and it is also seen that the PRNU pattern noise is clearly reminding to stay the same along different frames in the video, because results would be different in that case.

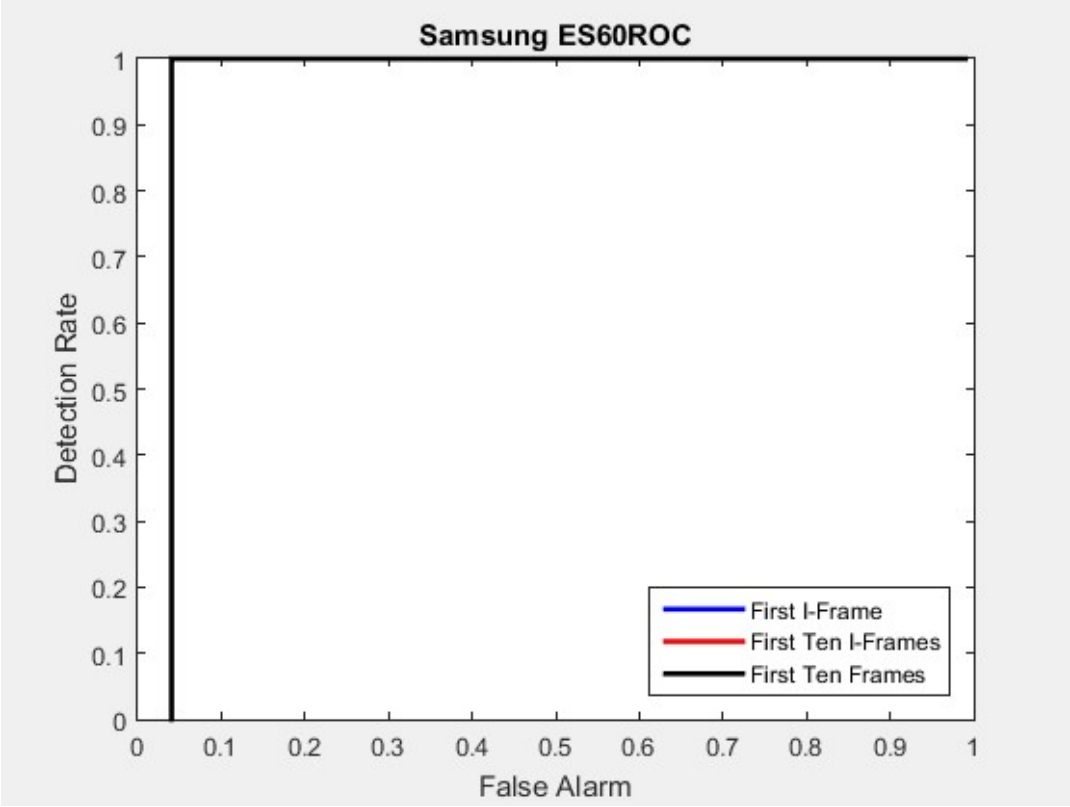

Image 4.4.2.12. ROC Curve for camera identification in Samsung ES60.

m) Samsung_GT-I9070P_Tiziano:

In the case of this last camera with a GOP structure of IPPPP… where the last two methods can be implemented on, we have an almost perfect detection (that is because we only have one natural video of this camera, but the detection with more videos should also be high).

There is not too much more to comment here, just that this curve is similar to the calculated in (10). There is achieved perfect detection in all methods except the first one, but is also high. It is the same case of this code, and as we can see in *CorrelationCorrectThresholding.xlsx* Excel project, for the first method correlation values between natural video Fingerprints and the Reference Fingerprints of the same camera is under 3 times the standard deviations of those correlation procedures (thresholds taken to consider well detected those with higher value than threshold), so that would be the only reason of the imperfection of this Fingerprint estimation process.

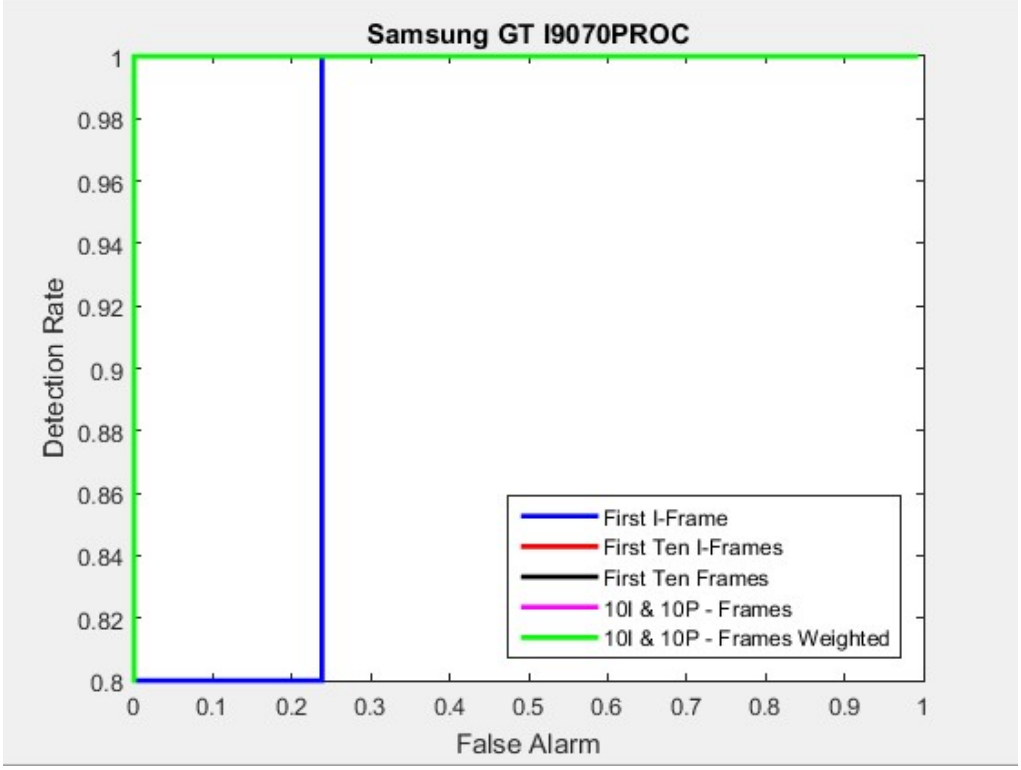

Image 4.4.2.13. ROC Curve for camera identification in Samsung GT I9070P.

4.5) Experiment 5: Further analysis of *Apple* devices.

As we can see in the previous ROC curve graphics in general for all devices with Motion JPEG except in those with low image quality the code works well detecting, and for the most cameras without Motion JPEG there is good detection but there are some exceptions. The most noticeable cases are the ROC curves obtained from detection in *Apple* devices, so for further analysis to try to understanding why those strange situations happen an experiment will be made unique for those devices. Attending to this, two main possibilities are proposed: Natural Fingerprints are bad estimated, or Reference Fingerprints of those devices are bad Estimated. Until this point of the thesis experiments are mainly focused in the Natural Fingerprints because of the analysis of the 5 different methods and their results on detection. So the proposed experiment to investigate those anomalies consists on checking the correlation between two different Reference Fingerprints of the same camera obtained from two different subsets of frames for each camera. If the correlation value between two Reference Fingerprints of the same camera is high, in that camera can be discarded the possibility of having a bad estimation of Reference Fingerprint. If the opposite situation occurs, there will be proved that the estimation of Reference Fingerprint is not good and in a big part would be the reason of the existence of those anomalies.

To carry on this experiment first we have to generate the 6 variables that contain the estimations of the 3 *Apple* camera Reference Fingerprints of two subsets of 10 I-frames for each camera. This task will be completed by the MATLAB script called *VIDEO_GetAppleRefFingerprints.m*, which automatically will generate those variables. After this a normal correlation will be computed between all variables, and for this process there is designed other MATLAB code called *VIDEO_CorrUncompressed_Apple.m*, which will return a 3x3 confusion matrix where the rows will refer to Reference Fingerprints obtained from first 10 I-frames for 3 cameras, and the columns their respective next 10 I-frames.

The result is illustrated in the next table:

	Apple_Azzurra_2	Apple_Giulia_2	Apple_Tiziano_2
Apple_Azzurra_1	0,00075	0,00022	0,000513
Apple_Giulia_1	0,00067	0,000993	0,00095
Apple_Tiziano_1	0,00133	-0.00008	0,004

Table 4.5.1. Confusion matrix of Reference Fingerprints of two different subsets of I-frames in Apple devices.

In this table there can be clearly seen how low the correlation values of the diagonal of this confusion matrix are, where should be high values. With this experiment is probed that the Reference Fingerprint is the main problem in the camera identification *Apple* devices. So, this bad estimation is in big part the origin of the bad values in the ROC curves for those devices. The main reasons may could be that the smooth videos are not bright enough or long enough. If videos are longer more I-frames could be taken and more accurate would be the Reference Fingerprint, and if videos are nor bright enough one solution would be to take other flat and smooth video of low variance but brighter in the sense of intensity of light. We can see in those results of the previous table how low are the values of correlation between two subsets of the same camera, even making sure that we are taking I-frames for the estimation.

5. Conclusions.

Up to this point, we have already explained several chapters as the state of art or the theoretical basis of this thesis, also all MATLAB codes used although they are developed from other previously given that belong to older thesis works, and the most important part of the thesis as well, the experiments done mainly to improve the older designs of camera identification algorithms.

As it is seen in the experimental results chapter we can mainly see the improvement of detection based on the obtained ROC Curves, and from there we concluded that there are several factors affecting the performance of camera identification from videos, such as the importance of having good image quality or resolution, of taking as much I-frames as possible, enough brightness, especially for *Apple* devices in this study having a proper estimation of Reference Fingerprint and the most important conclusion that it has supposed the biggest improvement in camera identification algorithm of this thesis, making sure that correct frames are taken for estimation without assuming any supposition or premise like all videos have fixed GOP structure.

In order of experiments, first of all a very basic but also needed experiment is done to obtain the GOP sizes of the different videos of the provided 13 cameras, just because is needed, as explained in 3.1.1 subchapter, this parameter as an input in the Fingerprint estimator function. After this, seeing the results obtained in previous work (10) we observe that probably those bad results in no Motion JPEG videos (there is GOP) come from there is not a fixed size so the method did not always employ the correct frames for estimating the fingerprint.

So the next experiments are based, as explained in the previous chapter, on probing this theory and if it is true try to solve it in order to obtain better results and turn this algorithm in one even better for camera detection. As brief resume, next experiments are based on the checking if correct sentence is taken for estimation based on the proposed estimation methods, and a checking method explained in 3.2.4 is proposed for each method except the one that takes first 10 frames.

Once probed in the experimental sections 4.2 and 4.3 that in some cases the previous method used a wrong set of frames, there is deduced that some cameras use different structure from expected, like variable GOP size as occur in some *Apple* device videos or cameras that make videos that do not start with I-frames as in the case of *Nikon_D3100_Diego* where the starting frame is a B-frame. The proposed next step is generating using functions 3.2.1 and 3.2.2 and executable MATLAB script 3.2.3 one information text file for each video where is written plenty of information of each frame including the frame type, and from those files generate as variables vector containing each frame. Using those variables for well estimating both Natural and Reference Fingerprints, correlation is executed and after that all ROC Curves are shown, where we can see that all cases have same or better results from (10). But from *Apple* devices bad and strange results are obtained, so for further analyze those cases one last experiment is proposed where there is concluded that in big part the main problem of those bad results come from bad estimations of Reference Fingerprints.

So for finishing we need to underline the very high importance of having good estimations of both Reference and Natural Fingerprints. Concluded premises are that source videos for estimating Reference Fingerprints should be flat, smooth and bright enough, and after that would be enough taking correct frames in the way is proposed in this thesis. As for Natural Fingerprints, from proposed 5 methods should be chosen those that use as much I-frames as possible choosing them in the correct frame choosing way proposed also, and apart from it the light intensity and image quality should be high so they can help to make algorithm better or more efficient.

6. Bibliography.

1. **Bylthe, Paul and Fridrich, Jessica.** *Secure Digital Camera.* Binghamton, New York : s.n., 2004.

2. *Forensic Camera Classification: Verification of Sensor.* **Khanna, Nitin, Mikkilineni, Aravind K. and Delp, Edward J.** 1, West Lafayette, Indiana 47907-2035 : s.n., 2009, Vol. 11.

3. **Goljan, Miroslav.** Digital Camera Identification from Images – Estimating False Acceptance Probability. [book auth.] Kim Hyoung-Joong, Stefan Katzenbeisser and Anthony T. S. Ho. *International Workshop on Digital Watermarking.* Busan, Korea : s.n., 2008.

4. **Lefèbvre, Frédéric, et al.** Image and Video Fingerprinting: Forensic Applications. [book auth.] Edward J. Delp III, et al. *Media Forensics and Security.* San Jose, CA : SPIE Proceedings, 2009.

5. *Methods for identification of images acquired with Digital cameras.* **Geradts, Zeno J., et al.** , Volmerlaan (Nederlands) and Kashiwanoha (Japan) : SPIE Proceedings, 2001, Vol. 4232. 10.1117/12.417569.

6. **Linkwitz Lab. [Online] [Cited: June 28, 2014.] http://www.linkwitzlab.com/dpp/A-D-conversion.htm.**

7. Wikipedia. [Online] [Cited: May 25, 2016.] https://en.wikipedia.org/wiki/Image_sensor.

8. *Compressed Fingerprint Matching and Camera Identification via Random Projections.* Valsesia, Diego, et al. 7, s.l. : IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, 2015, Vol. 10.

9. *Source Digital Camcorder Identification Using Sensor Photo Response Non-Uniformity.* Chen, Mo, et al. 65051G, Binghamton, NY 13902–6000 : SPIE Proceedings, 2007, Vol. 6505.

10. Ghassemi, Sina. *Compressed Sensing for Video Camera Identification.* Torino, Italy : s.n., 2015.

11. Bingham, Ella and Mannila, Heikki. Random projection in dimensionality reduction: Applications to image and text data. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.* Helsinki, Finland : ACM New York, 2001.